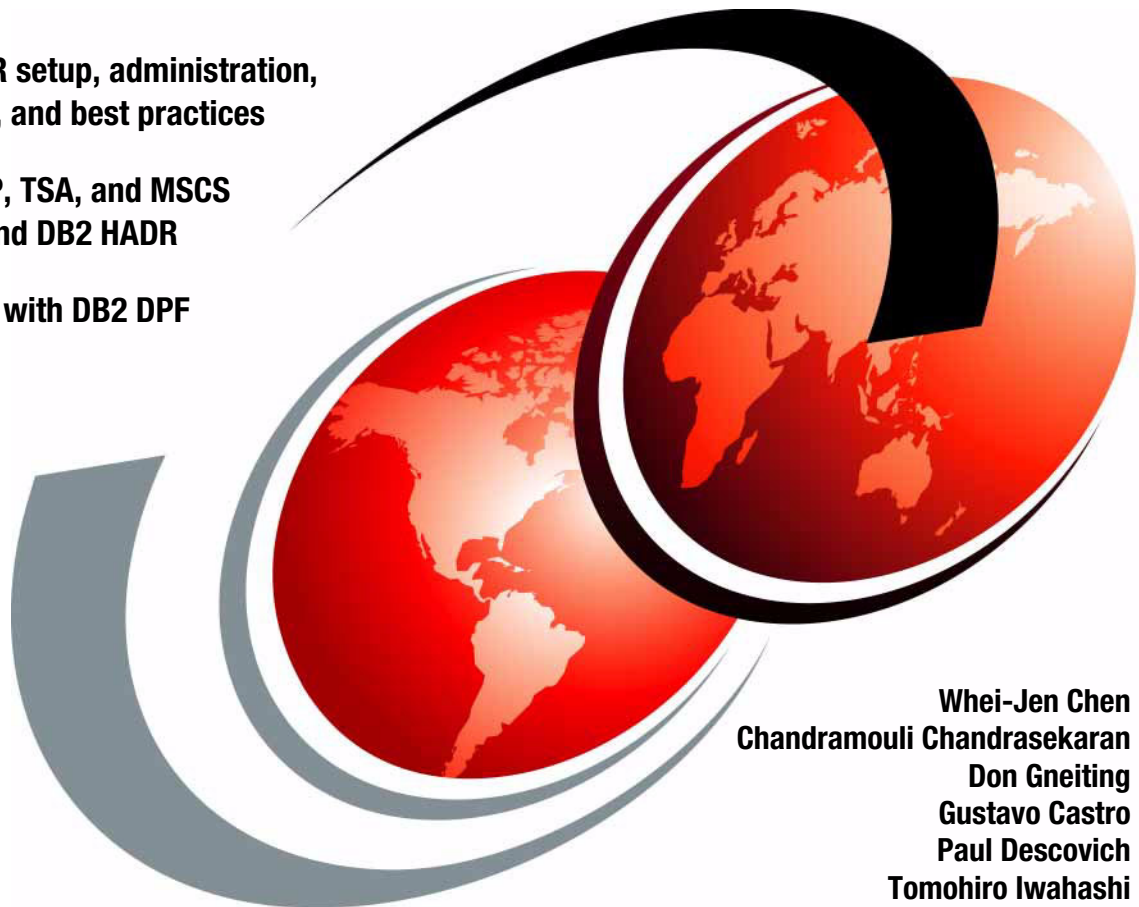


# High Availability and Scalability Guide for DB2 on Linux, UNIX, and Windows

Learn HADR setup, administration, monitoring, and best practices

Use HACMP, TSA, and MSCS with DB2 and DB2 HADR

Quick start with DB2 DPF



Whei-Jen Chen  
Chandramouli Chandrasekaran  
Don Gneiting  
Gustavo Castro  
Paul Descovich  
Tomohiro Iwahashi





International Technical Support Organization

**High Availability and Scalability Guide for DB2 on  
Linux, UNIX, and Windows**

September 2007

**Note:** Before using this information and the product it supports, read the information in “Notices” on page ix.

**First Edition (September 2007)**

This edition applies to DB2 for Linux, UNIX, and Windows, Version 9, and Version 8.2.

**© Copyright International Business Machines Corporation 2007. All rights reserved.**

Note to U.S. Government Users Restricted Rights -- Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.



# Contents

<b>Notices</b> .....	ix
Trademarks .....	x
<b>Preface</b> .....	xi
The team that wrote this Redbooks publication .....	xii
Acknowledgement .....	xiv
Become a published author .....	xiv
Comments welcome .....	xv
<b>Part 1. DB2 High availability options</b> .....	1
<b>Chapter 1. DB2 high availability options</b> .....	3
1.1 Introduction .....	4
1.2 Hardware-based solutions .....	4
1.2.1 RAID technology .....	5
1.2.2 Remote storage mirroring .....	9
1.3 Clustering solutions .....	10
1.3.1 Operating system dependent .....	11
1.3.2 Operating system independent .....	12
1.4 Replication .....	13
1.4.1 SQL replication .....	14
1.4.2 Q replication .....	15
1.4.3 DB2 HADR .....	17
1.5 Hybrid use of shared disk cluster and HADR .....	18
1.5.1 Overview .....	19
1.5.2 Case study of hybrid system .....	20
<b>Chapter 2. HADR introduction</b> .....	23
2.1 High availability overview .....	24
2.1.1 Solutions for high availability .....	24
2.1.2 Types of high availability .....	25
2.2 HADR overview .....	25
2.3 DB2 logical log architecture .....	28
2.4 HADR architecture .....	30
2.5 Terminology .....	36
<b>Chapter 3. HADR setup</b> .....	41
3.1 Requirements for setting up HADR .....	42
3.2 Setup and configuration .....	43

3.2.1	Preparing the environment . . . . .	43
3.2.2	Configuration using the Setup HADR wizard . . . . .	46
3.2.3	Command line setup . . . . .	65
3.3	Basic operation . . . . .	68
3.3.1	Starting up and shutting down. . . . .	68
3.3.2	Planned takeover . . . . .	73
3.3.3	Takeover by force . . . . .	77
3.4	Troubleshooting. . . . .	80
3.4.1	During setup . . . . .	81
3.4.2	After setup or during normal execution . . . . .	82
3.4.3	After an HADR disconnect or server failure . . . . .	84
3.4.4	Considerations while running HADR. . . . .	85
3.4.5	Re-establishing HADR after failure . . . . .	86
<b>Chapter 4. HADR administration and monitoring . . . . .</b>		<b>93</b>
4.1	Administering HADR systems . . . . .	94
4.2	Snapshot . . . . .	98
4.3	Monitoring HADR: db2pd . . . . .	101
4.4	Monitoring HADR: Administrative view and table function . . . . .	103
4.5	Monitoring HADR: db2diag . . . . .	107
4.6	Monitoring summary . . . . .	113
<b>Chapter 5. Automatic client reroute . . . . .</b>		<b>115</b>
5.1	Automatic client reroute overview . . . . .	116
5.1.1	DB2 automatic client reroute with HADR . . . . .	116
5.1.2	DB2 automatic client reroute in action . . . . .	117
5.2	Automatic client reroute tuning . . . . .	119
5.3	Limitations . . . . .	123
5.4	Examples. . . . .	124
5.4.1	Example 1: Involving a non-HADR database . . . . .	125
5.4.2	Example 2: Involving an HADR database . . . . .	127
5.4.3	Example 3: Involving an HADR database in HACMP cluster . . . . .	127
5.4.4	Sample application code to handle automatic client reroute. . . . .	129
<b>Chapter 6. HADR best practice . . . . .</b>		<b>137</b>
6.1	DB2 HADR configuration parameters . . . . .	138
6.1.1	Basic configuration parameters. . . . .	138
6.1.2	Automatic client reroute configuration parameters . . . . .	142
6.2	DB2 HADR registry variables . . . . .	143
6.3	Recommendations and considerations . . . . .	145
6.3.1	DB2 transaction performance . . . . .	145
6.3.2	DB2 utilities performance . . . . .	146
6.3.3	How to reduce the takeover time . . . . .	146
6.3.4	Seamless takeover . . . . .	147

6.3.5 Applications with high logging rate . . . . .	147
6.3.6 Performance implications of HADR_TIMEOUT . . . . .	147
6.3.7 Network considerations . . . . .	148
6.3.8 Avoiding data loss in an HADR with HA cluster software . . . . .	152
6.3.9 Index logging . . . . .	157
6.3.10 Read on the standby . . . . .	157
6.3.11 Backup from standby image with Flash Copy . . . . .	159
6.3.12 Replicating load data . . . . .	160
6.3.13 Log archive and HADR . . . . .	162
6.3.14 Database restore consideration . . . . .	162
6.4 Restrictions . . . . .	162
<b>Chapter 7. DB2 and system upgrades . . . . .</b>	<b>165</b>
7.1 DB2 FixPak rolling upgrades . . . . .	166
7.1.1 Applying DB2 FixPak on Windows operating systems . . . . .	166
7.1.2 Applying DB2 FixPak using command line . . . . .	172
7.2 Version upgrade . . . . .	173
7.2.1 Windows version upgrade . . . . .	174
7.2.2 Version upgrade using command line . . . . .	179
7.3 Rolling OS/Application/DB2 configuration parameters . . . . .	181
<b>Chapter 8. DB2 and HACMP . . . . .</b>	<b>183</b>
8.1 Overview . . . . .	184
8.2 How DB2 works with HACMP . . . . .	185
8.3 Planning the HACMP cluster . . . . .	188
8.4 Setting up HACMP . . . . .	190
8.4.1 HACMP cluster setup planning . . . . .	191
8.4.2 HACMP configuration . . . . .	193
8.5 Considerations for db2nodes.cfg file . . . . .	203
8.6 Tuning tips for quick failover . . . . .	208
8.6.1 Failure detection time . . . . .	208
8.6.2 Failover the resources . . . . .	209
<b>Chapter 9. DB2 with Microsoft Windows Cluster Server . . . . .</b>	<b>215</b>
9.1 Cluster Server concepts . . . . .	216
9.1.1 Types of clusters . . . . .	216
9.1.2 Microsoft Cluster Server definitions . . . . .	218
9.2 Minimum configuration and sample setup . . . . .	219
9.3 Creating a cluster server . . . . .	220
9.3.1 Adding the machines to the domain . . . . .	220
9.3.2 Creating the cluster in the domain . . . . .	223
9.4 Installing DB2 ESE . . . . .	231
9.5 Adding resources to a cluster for DB2 . . . . .	232
9.6 Creating a DB2 instance . . . . .	235

9.7	Manually configuring a DB2 instance in Windows Cluster	237
9.7.1	Adding the DB2 resource type	237
9.7.2	Creating a DB2 group	240
9.7.3	Creating a highly available IP address for the DB2 resource	242
9.7.4	Migrating the DB2 instance to the cluster environment.	244
9.7.5	Adding a reference to the instance in the other nodes	246
9.7.6	Creating the highly available DB2 instance resource	246
9.8	Configuring a highly available DB2 instance using db2mscs	248
<b>Chapter 10.</b>	<b>HADR with clustering software</b>	<b>253</b>
10.1	Overview: Why clustering software is needed.	254
10.2	Automating HADR takeover with HACMP.	258
10.2.1	HACMP and HADR planning	258
10.2.2	Step-by-step configuration overview	262
10.2.3	HADR setup	263
10.2.4	HACMP configuration	263
10.2.5	Prepare application server scripts.	276
10.2.6	Joint test for HADR and HACMP	279
10.3	Automating HADR takeover with TSA.	285
10.3.1	HADR with TSA.	290
10.3.2	Setting up HADR with TSA	293
10.3.3	Testing topology response to common failures.	308
<b>Chapter 11.</b>	<b>Q replication</b>	<b>315</b>
11.1	Introduction	316
11.2	Unidirectional setup.	317
<b>Part 2.</b>	<b>DB2 scalability options</b>	<b>355</b>
<b>Chapter 12.</b>	<b>DB2 Database Partitioning Feature</b>	<b>357</b>
12.1	Overview of DPF	358
12.1.1	DB2 partitioned instance.	361
12.1.2	The nodes configuration file	362
12.1.3	Catalog partition	363
12.1.4	Partition groups.	363
12.1.5	Distribution keys and distribution maps.	364
12.2	Installation	366
12.2.1	High level installation procedure	366
12.2.2	Verifying the installation	372
12.3	Scaling the database.	373
12.3.1	Balanced configuration unit.	373
12.3.2	Scaling the number of partitions	374
12.4	High availability on DPF	377

<b>Chapter 13. Table partitioning</b>	383
13.1 Introduction	384
13.2 Benefits of table partitioning	385
13.2.1 Efficient roll-in and roll-out.	386
13.2.2 Easier administration of large tables	386
13.2.3 Flexible index placement.	386
13.2.4 Data partition elimination during query processing	387
13.3 Creating a partitioned table	387
13.3.1 Prerequisites	387
13.3.2 Creating partitioned table examples	389
13.4 Rotating data in a partitioned table	390
13.5 Locking behavior on partitioned tables	392
13.6 Understanding index behavior.	394
13.7 Approaches to defining ranges on partitioned tables	398
13.7.1 Automatically generated ranges	399
13.7.2 Manually generated ranges.	400
13.8 Restrictions	402
 <b>Appendix A. HACMP application server scripts</b>	403
A.1 hadr_primary_takeover.ksh	404
A.2 hadr_primary_stop.ksh	406
A.3 hadr_monitor.ksh	409
 <b>Related publications</b>	411
IBM Redbooks	411
Other publications	411
Online resources	413
How to get IBM Redbooks	414
Help from IBM	414
 <b>Index</b>	415
 <b>Abbreviations and acronyms</b>	423



# Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not give you any license to these patents. You can send license inquiries, in writing, to:

*IBM Director of Licensing, IBM Corporation, North Castle Drive, Armonk, NY 10504-1785 U.S.A.*

**The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:** INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.


This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

## COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

# Trademarks

The following terms are trademarks of the International Business Machines Corporation in the United States, other countries, or both:

AIX®	HACMP™	Redbooks (logo)  ®
AIX 5L™	IBM®	RS/6000®
DB2®	Informix®	System p™
DB2 Universal Database™	MVS™	Tivoli®
ECKD™	OS/390®	WebSphere®
FlashCopy®	Redbooks®	z/OS®

The following terms are trademarks of other companies:

Snapshot, and the Network Appliance logo are trademarks or registered trademarks of Network Appliance, Inc. in the U.S. and other countries.

Java, Java Naming and Directory Interface, JDBC, JVM, J2EE, Solaris, Sun, Ultra, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Active Directory, Microsoft, Windows NT, Windows Server, Windows, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, Xeon, Intel logo, Intel Inside logo, and Intel Centrino logo are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product, or service names may be trademarks or service marks of others.



# Preface

This IBM® Redbooks® publication, intended for IT managers, IT architects, DBAs, application developers, and other data server professionals, offers a broad understanding of DB2® high availability and scalability options. The book is organized as follows:

- ▶ **Chapter 1** provides an overview of some high availability technologies and products that can help you increase the availability of the DB2 database and applications. We discuss some scenarios where these technologies are suitable for your enterprise.
- ▶ **Chapter 2** provides a general overview of the DB2 high availability feature, High Availability Disaster Recovery, including HADR architecture, terminologies, and HADR advantages.
- ▶ **Chapter 3** contains steps to set up a High Availability Disaster Recovery environment using both command line and Graphical User Interface. This chapter covers the basic operations, including START HADR, STOP HADR, and TAKEOVER HADR commands. It also provides troubleshooting tips.
- ▶ **Chapter 4** gives a description of administration tasks related to a High Availability Disaster Recovery system. The methods of acquiring the status of HADR are described, including get snapshot, db2pd, table functions, and db2diag.log.
- ▶ **Chapter 5** describes an important DB2 feature automatic client reroute (ACR). ACR enables a DB2 client application to recover from a loss of communications so that the application can continue its work with minimal interruption.
- ▶ **Chapter 6** discusses the DB2 database and database manager configuration parameters and registry variables which affect HADR. It also discusses the import factors to be considered when implementing a HADR solution to get the most optimum HADR performance.
- ▶ **Chapter 7** focuses on how to perform DB2 FixPak applies with the no-downtime benefit which HADR brings, and DB2 version upgrades or migrations with HADR databases. This chapter also explores the considerations when performing changes to other system components or updating certain database configuration parameters.
- ▶ **Chapter 8** introduces how to integrate DB2 in a High Availability Cluster Multi-Processing (HACMP™) environment. It provides the basic management concepts with recommendations and considerations to reduce the time consumed for failover.

- ▶ **Chapter 9** provides the steps to set up a high available DB2 environment with Microsoft® Windows® Cluster. It also provides the fundamental concepts of Microsoft Windows Cluster Server architecture.
- ▶ **Chapter 10** discusses how to configure HADR with HACMP or TSA to enable automating HADR takeover. It includes the setup procedures of automating HADR takeover with HACMP and TSA.
- ▶ **Chapter 11** gives an introduction to Q replication and a simple overview of the structure. This chapter provides step-by-step instructions for setting up unidirectional Q replication.
- ▶ **Chapter 12** introduces the Database Partitioning Feature, which enables DB2 to scale out to support very large databases and increased parallelism for various tasks. This chapter includes the procedures of installing and scaling DB2 databases.
- ▶ **Chapter 13** discusses the improved large database management feature: table partitioning, introduced in DB2 9. This new feature allows DBAs to divide table data into partitions and stored in one or multiple table spaces. This new feature provides easier management, improved performance, and greater scalability for large databases.

## The team that wrote this Redbooks publication

This book was produced by a team of specialists from around the world working at the International Technical Support Organization, Poughkeepsie Center.

**Whei-Jen Chen** is a Project Leader at the International Technical Support Organization, San Jose Center. She has extensive experience in application development, database design and modeling, and DB2 system administration. Whei-Jen is an IBM Certified Solutions Expert in Database Administration and Application Development as well as an IBM Certified IT Specialist.

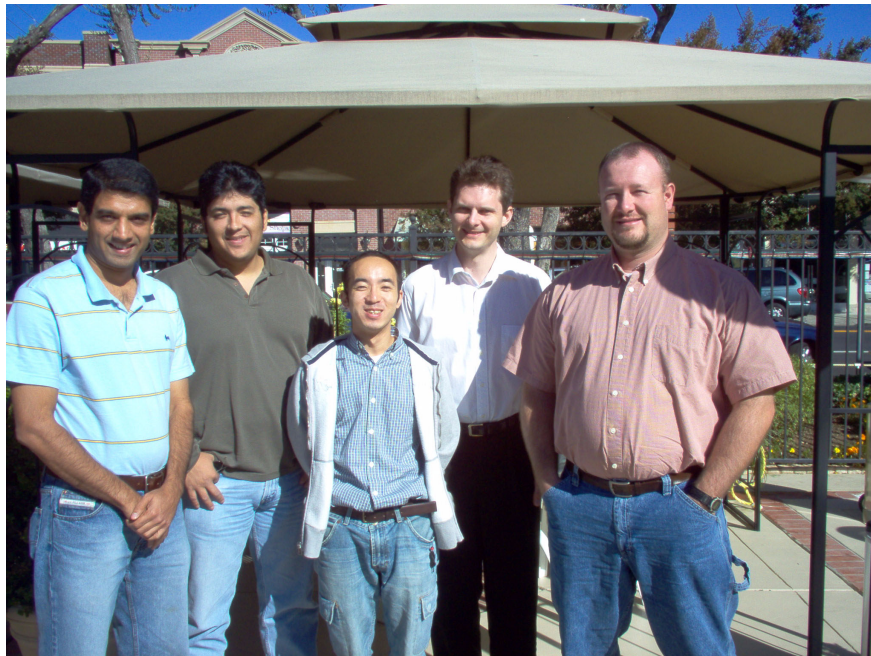
**Chandramouli Chandrasekaran** is a consultant at the IBM Toronto Lab team, where DB2 UDB for Linux®, UNIX®, and Windows is developed. He is a member of the Data Management Partner Enablement organization where he works with IBM business partners to migrate applications to DB2 and perform performance tuning. Chandra is a DB2 expert in both DB2 administration and application development.

**Don Gneiting** is an IBM Certified Database Administrator for IBM Integrated Technology Delivery, Server Systems Operations. He holds a bachelor's degree in Business Information Systems, as well as, a bachelor's in Finance from Utah State University, Utah. He has five years experience as a production system DBA.

**Gustavo Castro** is an Advisory Software Engineer. He has worked for IBM Corporation the last 5 years as an Advanced Support Engineer for Information Management products including Informix® and DB2 database engines in the areas of Performance, Backup/Recovery, and crash analysis.

**Tomohiro Iwahashi** is an IT Specialist. He has worked for IBM Japan for eight years designing and implementing AIX® systems for mainly Japanese financial industry. Tomohiro currently is the technical support for DB2 on Linux, UNIX, and Windows platforms in IBM Japan Systems Engineering Co.,Ltd, focusing on High Availability and scalability of database systems.

**Paul Descovich** works in a DB2 Systems Support role for IBM Global Services Australia, where most of his time is spent installing, configuring, and maintaining DB2 products. The rest of his time is spent motivating architects to determine how DB2 should be implemented on new servers. Paul started out in application programming for DB2 version 2 release 2 on MVS™ in 1991, then DB2 System Support and DBA work from 1993, joining IBM in 1999. Since 2000, his work has become almost exclusively related to DB2 on Linux, UNIX, and Windows platforms.



*Author team: Left to right - Chandra, Gustavo, Tomohiro, Paul, and Don*

## Acknowledgement

Thanks to the following people for their contributions to this project:

Deb Jenson  
Dwaine Snow  
Yuke Zhuge  
Steve Pearson  
Vincent Kulandaisamy  
IBM Software Group

Jessica Escott  
Budi Surjanto  
Belal Tassi  
Makiko Tsuchida  
Lok Chan  
Lui Tang  
IBM Toronto Laboratory

Tadakatsu Azuma  
Tetsuya Shirai  
Maki Nakayama  
Hiroyasu Minami  
Makiko Tsuchida  
Naoko Takaya  
Tomoko Ichikawa  
Toshihiko Kubo  
IBM Japan

Yvonne Lyon  
Sangam Racherla  
Deanna Polm  
Emma Jacobs  
Marcus Thodal  
Mary Lovelace  
Charlotte Brooks  
International Technical Support Organization, San Jose Center

## Become a published author

Join us for a two- to six-week residency program! Help write a Redbooks publication dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You will have the opportunity to team with IBM technical professionals, Business Partners, and Clients.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you will develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

[ibm.com/redbooks/residencies.html](http://ibm.com/redbooks/residencies.html)

## Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

[ibm.com/redbooks](http://ibm.com/redbooks)

- Send your comments in an e-mail to:

[redbooks@us.ibm.com](mailto:redbooks@us.ibm.com)

- Mail your comments to:

IBM Corporation, International Technical Support Organization  
Dept. HYTD Mail Station P099  
2455 South Road  
Poughkeepsie, NY 12601-5400





# Part 1

## **DB2 High availability options**

In this part of the book, we introduce the various high availability options that DB2 offers. We focus on the DB2 High Availability and Disaster Recovery (HADR) feature. We provide a detailed description of HADR, including setup and configuration, administration and monitoring, considerations, and best practices.

We describe the clustering software HACMP, TSA, and Microsoft Windows Clusters, explaining how to set up these product with DB2, and how to use them to automate the HADR takeover.

DB2 automatic client reroute is also discussed.







# DB2 high availability options

In this chapter we provide an overview of different high availability and fault tolerance options in the market, where we briefly describe some technologies and products that can help you increase the availability of your DB2 applications.

We review these technologies and discuss, at a high level, how you can integrate DB2 with them. We also discuss some scenarios where these technologies might be suitable for your enterprise.

## 1.1 Introduction

DB2 provides several features that can increase high availability of the database servers, including these:

- ▶ High Availability and Disaster Recovery (HADR)
- ▶ Automatic Client Reroute (ACR)
- ▶ SQL replication
- ▶ Q replication
- ▶ Some basic DB2 features also increase the availability of your operation, such as online backup and in-place reorganization.

In addition to these options, DB2 supports a number of software and hardware offerings from IBM and other providers that you can use in conjunction with DB2 to strengthen high availability in your environment.

The decision on what products or features to use depends, as always, on the specific challenges of the environment, budget, complexity, time to implement, and degree of security required.

DB2 offers cost-effective alternatives to implement a very reliable solution. You can replace some of the DB2 high availability features with additional hardware and software, which, most of the time, drives costs up. For example, most high availability solutions implemented at the operating system level (clusters) heavily rely on shared disks, making storage the weak link in your operation. To overcome this, you are required to implement storage mirrors that escalate costs.

## 1.2 Hardware-based solutions

Hardware solutions usually have the highest performance and security by copying or replicating whole disks. Using a hardware solution reduces the granularity and ability to determine when and what must be replicated. These solutions are also usually much more expensive than software solutions.

DB2 offerings are completely integrated with these hardware-based type of solutions. The redundant array of independent disks (RAID) technology and remote storage mirroring are commonly used in a database system.

## 1.2.1 RAID technology

Redundant array of independent disks (RAID) technology consists of a group of individual disks that can be accessed as a whole to increase performance, fault tolerance, or both. While RAID technologies can be implemented at the software level, such implementations usually do not have the level of security or performance most database servers require. In this chapter, we discuss only hardware level RAID.

### RAID 0

RAID 0 is also known as *disk striping*. It refers to the usage of two or more disks, where the disk controller (known as a RAID controller) splits the I/O operations among the drives, creating a parallel write or read operation that increases performance but not reliability.

As Figure 1-1 shows, the RAID controller receives an 8 KB block to write. The controller splits the data into four 2 KB and writes the data parallel among all drives. RAID 0 provides no redundancy.

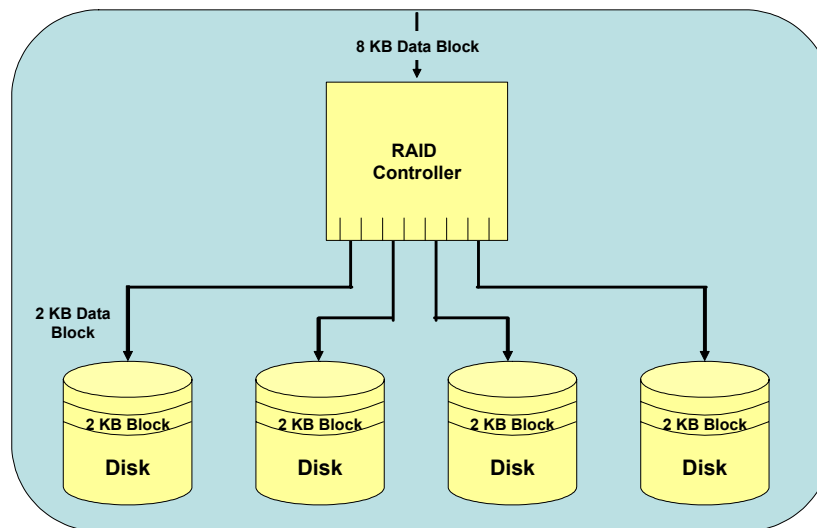


Figure 1-1 RAID 0 (disk striping) architecture

### RAID 1

RAID 1 is also known as *disk mirroring*. In RAID 1, the RAID controller replicates the writing operations from the main (or primary) disk to one or more spare disks called *mirrors*. Performance of write operations does not suffer, as the writing to the devices is accomplished in parallel.

Read operations are actually improved as the controller uses all devices to read from, reducing contention as illustrated in Figure 1-2.

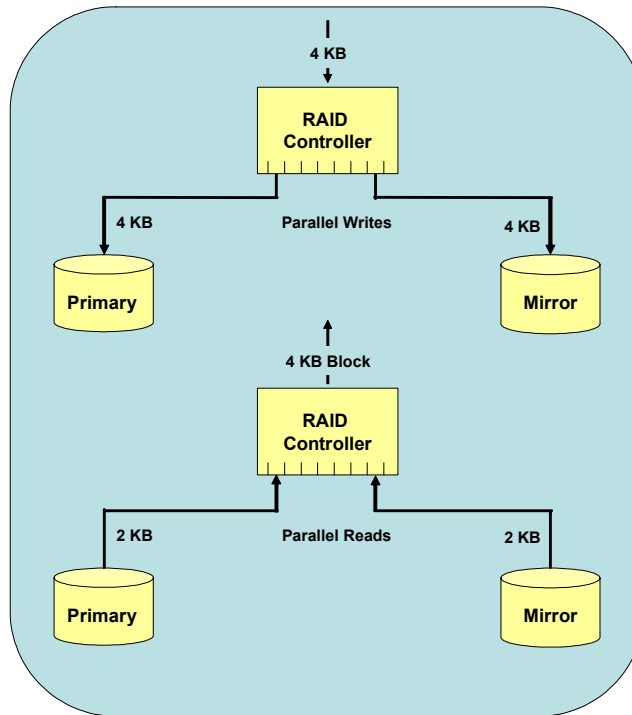


Figure 1-2 RAID 1 (disk mirroring) architecture

### RAID 3

Mirroring is great for security and very good for read performance, but it requires twice as much disk, which can be expensive. RAID 3 provides an alternative.

With RAID 3, a minimum of three disks is required. One of these disks is dedicated to store parity information about the data of the other two. In the event of a disk failure, the information in any of the disks can be re-created from the other two disks.

The downside of this technology is that the parity disk receives a heavy load, as any write operation requires a write on the parity disk. As illustrated in Figure 1-3, all writes require the computation of parity information, which is an additional CPU overhead. The more disks you have, the lower the percentage of disk space dedicated for parity.

Note that RAID 3 is not recommended for database storage, because of the single parity disk bottleneck.

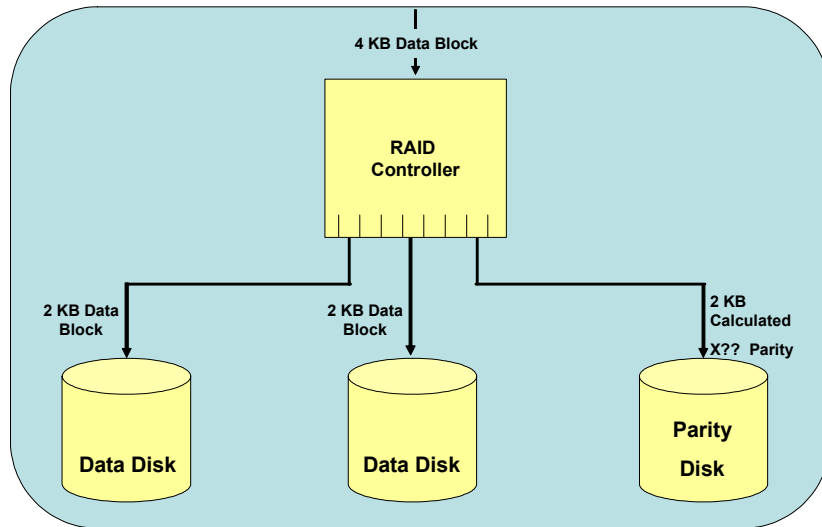


Figure 1-3 RAID 3 architecture in implementation

## RAID 5

RAID 5 is a variation of RAID 3. It has the same concept but instead of dedicating one single disk for parity, it spreads the parity among all the devices to balance the write load.

RAID 5 is a cost-effective alternative to mirror data. It provides redundancy at the expense of one physical disk per array and four I/O (2 reads and 2 writes) operations per I/O write request. As the number of physical disks in the array increases, the cost penalty for the array decreases, but the write penalty remains the same. Therefore, RAID 5 is not an optimal choice for online transaction processing (OLTP) database or random write I/O intensive systems.

Figure 1-4 shows how a basic RAID 5 system works.

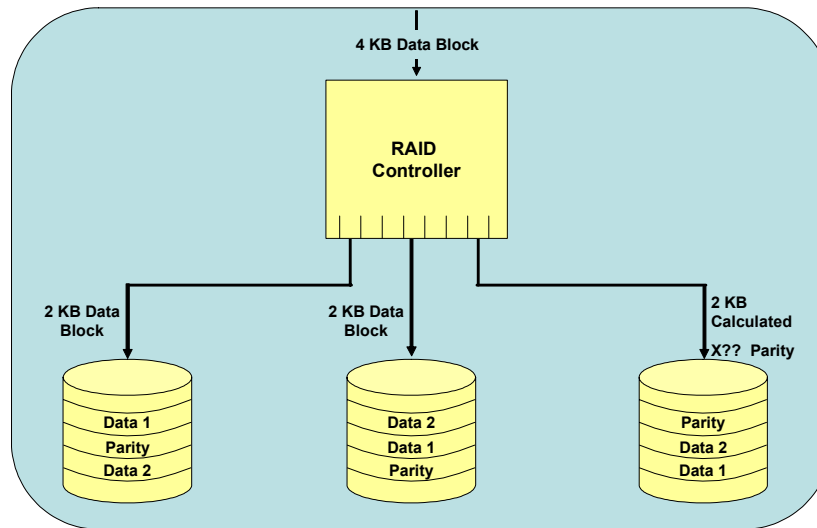


Figure 1-4 A basic RAID 5 example system

## RAID 10 and RAID 01

RAID 10 and RAID 01 (also known as 1+0 and 0+1) both use a combination of mirroring with striping. This is a winning combination for performance and fault tolerance but can be quite expensive.

The difference between RAID 10 and RAID 01 is that with RAID 10, individual disks are mirrored, and these mirrored groups are striped. On the other hand, with RAID 01 the mirror is applied to a whole striped volume. In other words, RAID 10 is a “stripe of mirrors”, and RAID 01 is a “mirror of stripes”. Because the failure of a single element in the stripe makes it unusable, the best alternative is to always stripe mirrored volumes, that is RAID 10.

From a pure performance and fault tolerance point of view, RAID 10 (1+0) is the perfect match and the best option for databases. RAID 5 can be a very cost-effective alternative for non-OLTP database storage.

## When to use

In general, RAID can be considered for the following situations:

- ▶ When there are no concerns about site failures; this means, where the entire RAID would be lost or taken offline
- ▶ When fault tolerance, performance, or both are the most important factors

## DB2 integration

DB2 databases can benefit from using RAID. As mentioned before, RAID 5 is not suitable for databases running OLTP applications. However, using RAID 1 or RAID 10 for tablespaces increases the performance, tolerance to failure of your application, or both.

For more information about RAID, see the Web site:

[http://en.wikipedia.org/wiki/Redundant\\_array\\_of\\_independent\\_disks](http://en.wikipedia.org/wiki/Redundant_array_of_independent_disks)

### 1.2.2 Remote storage mirroring

For years, mirroring and RAID have been the tools of choice to increase availability and performance at the hardware level. With ever increasing demands, many enterprises must not only be sure that their operations do not stop if a site failure occurs, but also require fast access to data at a remote location. Remote storage mirroring combines mirroring and replication in enterprise-level storage, allowing enterprises to provide solutions for these and other requirements.

The concept behind this technology is to have two or more disk arrays in boxes (also referred to as *Storage Enclosures*), located in geographically remote locations. These boxes are connected using *Dark fiber* or similar technologies and implement algorithms to replicate the data between the boxes.

Storage mirroring replicates all disk writes, including the log information. There are specific ways to update the remote disk change from brand to brand. In general, synchronous and asynchronous are two typical modes to update the remote disk.

Most of the time, remote storage mirroring has to be integrated with additional resources if automatic failover operations are required. Some examples of remote storage mirroring products are:

- ▶ IBM Peer to Peer Remote Copy (PPRC) and IBM Peer to Peer Remote Copy over eXtended Distances (PPRC-XD)
- ▶ Hitachi TrueCopy

#### Advantages

The advantages of remote storage mirroring are as follows:

- ▶ Several copies of data are kept in geographically remote locations.
- ▶ Copies are made at the hardware level, which usually improves performance and reliability.

## When to use

Use this technology when no data loss can be tolerated and the copy must be made with minimum delay.

## DB2 integration

Remote storage mirroring seamlessly integrates with DB2, allowing the enterprise to have a disaster recovery site providing continuous service. Additional technology, such as clustering, is required to automate the process of starting DB2 at the new site.

For more information, refer to the following Web sites:

- ▶ Dark fiber:

[http://en.wikipedia.org/wiki/Dark\\_fiber](http://en.wikipedia.org/wiki/Dark_fiber)

- ▶ PPRC:

[http://en.wikipedia.org/wiki/Peer\\_to\\_Peer\\_Remote\\_Copy](http://en.wikipedia.org/wiki/Peer_to_Peer_Remote_Copy)

## 1.3 Clustering solutions

Clustering is a technique that has been employed for a long time to increase the resilience, performance, and availability of the solutions implemented in servers. A *cluster* is a virtual machine made up of two or more physical machines (or nodes), each with various subcomponents registered to the cluster configuration as resources. The functionality of the resources and the nodes is logically bound together using additional software and hardware. Clusters create the concept of highly available resources. In the event of a node failure, resources on other nodes can take over the workload, minimizing the downtime.

The idea of clustering is to present to the users a single machine, when in fact the system has multiple nodes to serve the client applications. Many clusters act in an active/passive configuration where only one node performs work, with the others standing by as the backup in case of failure. Some cluster solutions are sophisticated enough to allow load balancing between the nodes in an active/active configuration, thus maximizing the performance of the applications, and providing more cost-effective use of the resources.

Clustering solutions can be broadly categorized in to two types:

- ▶ Operating system dependent
- ▶ Operating system independent



### 1.3.1 Operating system dependent

Most operating systems provide tools or products to create clusters. The techniques and concepts are similar, and the intention is always to present to the world a single virtual server, although the services can be spread among the nodes or members of the cluster.

In the event of a service or resource failure, cluster processes try to restart the resource in the affected node, and if that is not possible, then resources on another node take their place.

Irrespective of cluster implementation, connection-oriented applications, such as database applications, require embedded logic to detect lost connections and to retry those operations that were in doubt or unfinished. The advantage of the cluster is that the clustering software often provides features for the applications to continue working effectively with no transactional data loss when the transfer of services between the nodes is automated. This reduces downtime and human intervention.

These are some of the existing cluster solution providers:

- ▶ IBM HACMP for AIX and Linux for IBM System p™
- ▶ Microsoft Windows Server® 2003 Clustering Services for 32-bit/64-bit Intel/AMD platforms (formerly MSCS) and Windows Compute Cluster Server 2003 for 64-bit Intel/AMD platforms.
- ▶ HP Serviceguard for HP-UX and Linux
- ▶ Sun™ Cluster for Solaris™

#### **Advantage**

The advantage of the operating system dependent clustering software is that it is highly integrated with the operating system.

#### **When to use**

If you have a homogeneous architecture, operating system dependent clustering software that is well integrated with the operating system is a good choice.

#### **DB2 integration**

DB2 supports all the operating system dependent clustering software listed above. There are DB2 agents or resource types to integrate these cluster solutions with DB2. The integration is either provided from DB2 as libraries that are automatically installed, or as sample scripts.

For more information about operating system dependent software, see the following references:

- ▶ IBM HACMP:  
[http://en.wikipedia.org/wiki/High\\_Availability\\_Cluster\\_Multiprocessing](http://en.wikipedia.org/wiki/High_Availability_Cluster_Multiprocessing)
- ▶ Microsoft Windows Server 2003 Clustering Services:  
<http://www.microsoft.com/windowsserver2003/technologies/clustering/default.aspx>
- ▶ Microsoft Windows Compute Cluster Server 2003:  
<http://technet2.microsoft.com/windowsserver/en/technologies/featured/ccs/default.aspx>  
<http://www.microsoft.com/windowsserver2003/ccs/default.aspx>
- ▶ HP Serviceguard:  
<http://h71028.www7.hp.com/enterprise/cache/6469-0-0-0-121.html>
- ▶ Sun Cluster:  
<http://www.sun.com/software/cluster/>

### 1.3.2 Operating system independent

These solutions have characteristics similar to the operating system dependent clustering software. This type of clustering software can work with various operating systems. They usually provide more sophisticated features to manage and control the clusters. Some allow frameworks which manage and interchange resources among groups of clusters.

Because these solutions are highly specialized, sometimes they offer more functionality than their operating system dependent counterparts. Some of these tools are:

- ▶ Tivoli® System Automation (TSA): End-to-End Automation and Base components
- ▶ Veritas Cluster Server

#### **Advantages**

The operating system independent clustering software has the following advantages:

- ▶ These solutions are platform independent.
- ▶ They offer a single interface across different platforms.

## When to use

You can use the operating system independent clustering software:

- ▶ When you have a heterogeneous environment with different hardware providers.
- ▶ If platform change is a possibility.
- ▶ If your OS does not offer a cluster solution or has restrictions that can be avoided with an operating system independent clustering software.
- ▶ If you require to centrally manage a complex set of existing base-level clusters on differing platforms, which individually comprise only part of a larger overall application, then a multi-tiered cluster management solution such as TSA End-to-End Automation can plug in to existing base-level clusters without any requirement to swap architecture or vendors.

## DB2 integration

There are DB2 agents or resource types to integrate these cluster solutions with DB2. All previously mentioned solutions support DB2. The integration is always provided by the cluster tool.

For more information on operating system independent software, refer to the following Web sites:

- ▶ Tivoli System Automation:  
<http://www-306.ibm.com/software/tivoli/products/sys-auto-multi/>
- ▶ Veritas Cluster Server:  
[http://en.wikipedia.org/wiki/Veritas\\_Cluster\\_Server](http://en.wikipedia.org/wiki/Veritas_Cluster_Server)

## 1.4 Replication

Replication solutions have been in use for quite some time. There are multiple replication technologies to suit different environments. In this section we focus on software-based replication.

DB2 Universal Database™ for Linux, UNIX, and Windows provides two different solutions that you can use to replicate data from and to relational databases: SQL replication and Q replication. In SQL replication, committed source changes are staged in relational tables before being replicated to target systems. In Q replication, committed source changes are written in messages that are transported through WebSphere® MQ message queues to target systems.

DB2 also provides a solution called *event publishing* for converting committed source changes into messages in an XML format and publishing those messages to applications such as message brokers.

### 1.4.1 SQL replication

SQL replication is integrated in DB2 using IBM WebSphere Information Integration.

The source system has a capture program that scans the DB2 database log files looking for records belonging to replicated tables and then places this information in temporary storage known as *staging tables*.

The records in these staging tables are sent to the target system, where an apply program executes them as DB2 SQL operations against the replica database.

Figure 1-5 shows an overview of SQL replication architecture. Inside the source server are a set of DB2 relational tables called *Capture control tables*. Information about sources is written to these tables. The Capture program, which runs on the source server, uses this information to know what data it is supposed to capture. Information about targets goes in the Apply control tables, which are typically on your target server. The Apply program, which is also typically on your target server, uses this information to know which targets it is supposed to write data to.

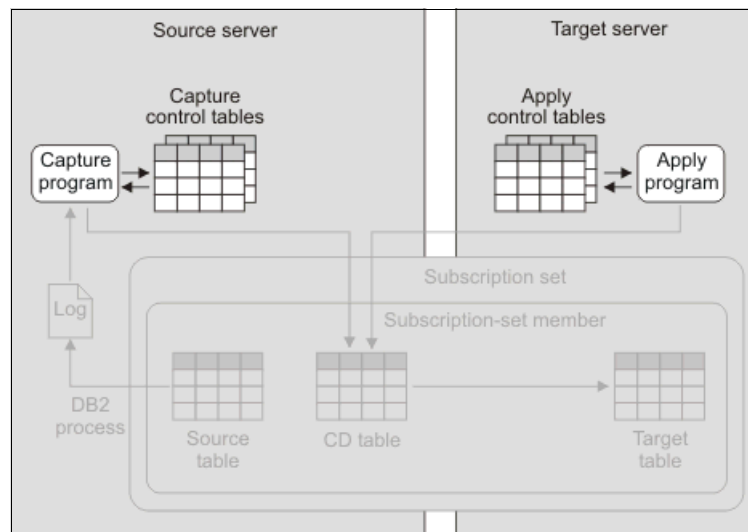


Figure 1-5 Infrastructure for a simple configuration in SQL replication

## Advantages

These are some advantages of DB2 SQL replication:

- ▶ It includes DB2 for OS/390® and z/OS® sources, targets, or both, in addition to DB2 for Linux, UNIX, and Windows sources, targets, or both.
- ▶ It can access multiple database management system (DBMS) types of sources and destinations.
- ▶ Replica databases are not disabled or hidden from client connectivity  
Client read-only access to replica database is recommended with SQL replication due to unidirectional replication.

## When to use

You can use SQL replication in the following situations:

- ▶ If you have a heterogeneous database provider environment, SQL replication can access more source and destinations than Q replication.
- ▶ When you have multiple targets from a single source, the information can be replicated or broadcast from the staging tables. This can suit geographic distribution for read-only or aggregate data for Management Information Systems or data warehousing.

For more information on SQL replication, refer to the following sources:

- ▶ *IBM WebSphere Information Integration: SQL Replication Guide and Reference Version 9*, SC19-1030-00
- ▶ SQL replication: overview:  
[http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod\\_overview/iiyrcintch200.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iiyrcintch200.html)
- ▶ Replication solutions for common scenarios:  
[http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod\\_overview/iiyrcintrsbdd.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iiyrcintrsbdd.html)

### 1.4.2 Q replication

Q replication is a replication solution that uses message queues in WebSphere MQ to transmit transactions between source and target databases.

The Q Capture program runs on the source server, and reads DB2 recovery logs for changes in the source data. These changes are then written to the WebSphere MQ queues. The Q apply program runs on the target server and receives the changes from the queues and writes to the targets.

Q replication is divided into three types:

- ▶ Unidirectional:
  - One-way replication from a source to a target
  - Possibility to configure multiple sources to a target or multiple targets to a source
- ▶ Bidirectional:
 

Two-way replication between a source and a target table on two servers
- ▶ Peer to peer:
 

Two-way replication between a source and a target table on two or more servers

Figure 1-6 describes a unidirectional architecture using Q replication. Note that reference is only made to a DB2 process on the source server; however, this does not preclude client connectivity on the target server. DB2 processes exist here with an active target database in order to accept the SQL transactions being converted from message queue format by the Q Apply program.

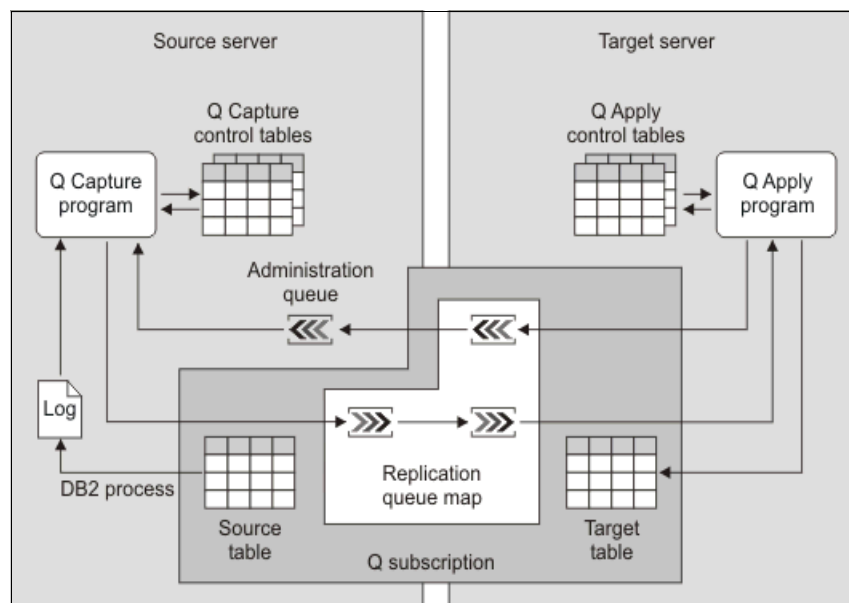


Figure 1-6 A simple configuration in Q replication

## Advantages

The advantages of Q replication are as follows:

- ▶ DB2 for OS/390 and z/OS sources, targets or both, in addition to DB2 for Linux, UNIX, and Windows sources, targets or both
- ▶ Supports high volumes of changes
- ▶ Low latency requirements
- ▶ Supports a large number of source/target tables
- ▶ Replicas/targets with conflict detection/resolution that can be updated
- ▶ Captured transactions can feed other applications

## When to use

You can use Q replication:

- ▶ For capacity relief such as read/write database for both source and target, splitting the transaction load between the servers with an external load balance mechanism.
- ▶ Geographically distributed applications to update back to a central database or in the opposite direction with a central hub broadcasting to multiple distributed targets. Note that we recommend SQL replication for the latter configuration.
- ▶ For rolling upgrades.
- ▶ When a subset of columns and rows are required for replication.

For more information refer to Replication Scenarios for Common Solutions on the Web at:

[http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod\\_overview/iiyrcintrsbdd.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.replication.intro.doc/prod_overview/iiyrcintrsbdd.html)

## 1.4.3 DB2 HADR

DB2 High Availability and Disaster Recovery (HADR) is a feature in DB2 Enterprise Server allowing replication of any logged database activity to a local or remote location.

DB2 HADR works in a similar fashion to the storage mirroring solutions, but all the work is made inside DB2 at the software level. A DB2 HADR primary database uses internal processes to ship database logs to an HADR standby database. A process at the standby server then replays these logs directly to the standby database. The standby database can be switched online in the event of a disaster, or whenever the primary requires to be temporarily taken offline for scheduled downtime.

## Advantages

The advantages of using DB2 HADR include:

- ▶ Ultra™ fast failover capability
- ▶ Negligible impact on performance
- ▶ Easy to set up and monitor
- ▶ Rolling upgrades without service interruption between non-major versions or changes requiring server recycling, and reduced change windows for other upgrades
- ▶ Transparent failover and failback for applications
- ▶ Easy integration with high availability (HA) clustering software
- ▶ Dramatically improved disaster recovery compared to conventional methods

## When to use

You can use DB2 HADR:

- ▶ When there is a requirement for a disaster recovery site with fast failover.
- ▶ No committed transaction data loss, where the replica (standby) database must be continually kept up-to-date with the primary. This means that you have the benefits of two-phase commit between separate systems, without the crippling performance overheads of synchronous update activity.
- ▶ No downtime for system change windows. HADR allows the rolling DB2 FixPak applications and certain configuration parameter changes to both databases, and operating system maintenance/recycling windows with no downtime for DB2 clients.

For more information on DB2 HADR, refer to:

*DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228.

## 1.5 Hybrid use of shared disk cluster and HADR

For a local HA solution, a disk shared cluster with clustering software is the most well known configuration. HADR provides an HA solution that can be used in a database server that is physically in the same room or remotely. This section describes the combination (hybrid) of these two solutions and how this hybrid system strengthens high availability of the database system.



## 1.5.1 Overview

There are several considerations for a shared disk cluster. Because resources are shared between the cluster nodes, a disk shared configuration cannot provide services in the following situations:

- ▶ Crash or planned maintenance of the shared storage device.
- ▶ Maintenance requires to stop the cluster software.
- ▶ FixPak update or configuration change that requires to stop database instance.
- ▶ Site disaster

By adding HADR standby database to the shared disk cluster, continuous service is possible even in the cases mentioned previously. Here we call this system a *hybrid system*. Figure 1-7 illustrates a hybrid high availability system with shared disk cluster and HADR.

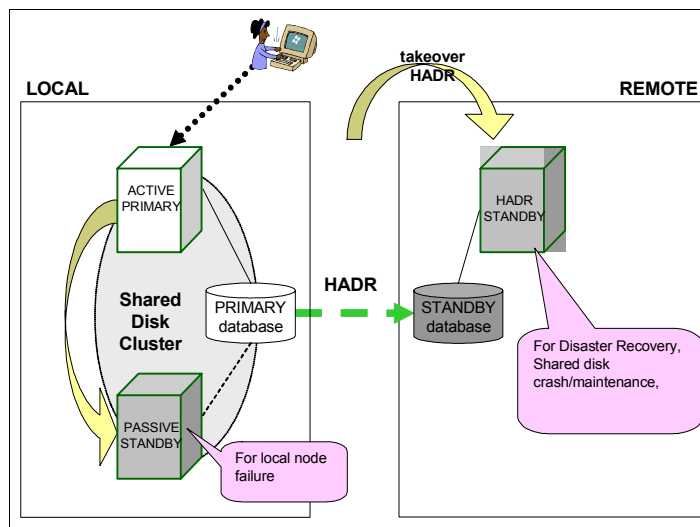


Figure 1-7 Hybrid high availability system with shared disk cluster and HADR

In this system configuration, there are three nodes with different roles:

- ▶ Active primary node:

This node provides service with shared disks. DB2 instance is running on it. Also, it plays a role of HADR primary database, transferring log records to standby database.

- ▶ **Passive (local) standby node:**  
If the active primary node fails, the local standby node takes over the resources, including shared disk.
- ▶ **HADR (remote) standby node:**  
The node where the HADR standby database is running. When both active primary node and the passive standby node are not available, this node can provide service. HADR standby can be placed in the same local site as the other nodes, but it must be in a remote site for the purpose of disaster recovery.

## 1.5.2 Case study of hybrid system

The following are the operations required of a hybrid system when possible failure or maintenance in the database system occurs:

- ▶ **Failure of the primary node:**  
The resource group, including the shared disk, is automatically failed over to the passive standby node, and the service is restarted. After failover, the HADR connection is re-established with the new active primary (passive standby), as shown in Figure 1-8.

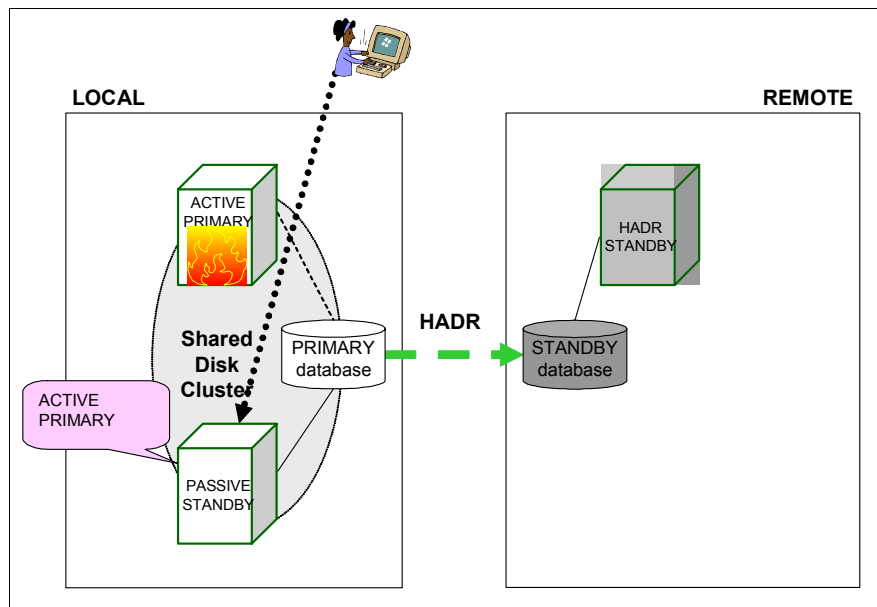


Figure 1-8 Failure of the primary node

► Crash or planned maintenance of the shared storage device:

If the shared disk is unavailable for any reason, the shared disk cluster cannot continue its services. In this case, the service can be taken over to the HADR standby. Even when the shared disk has to be stopped for the purpose of maintenance, such as upgrading the microcode of storage devices, or for restructuring the storage configuration. You do not have to stop database services for applications, as shown in Figure 1-9.

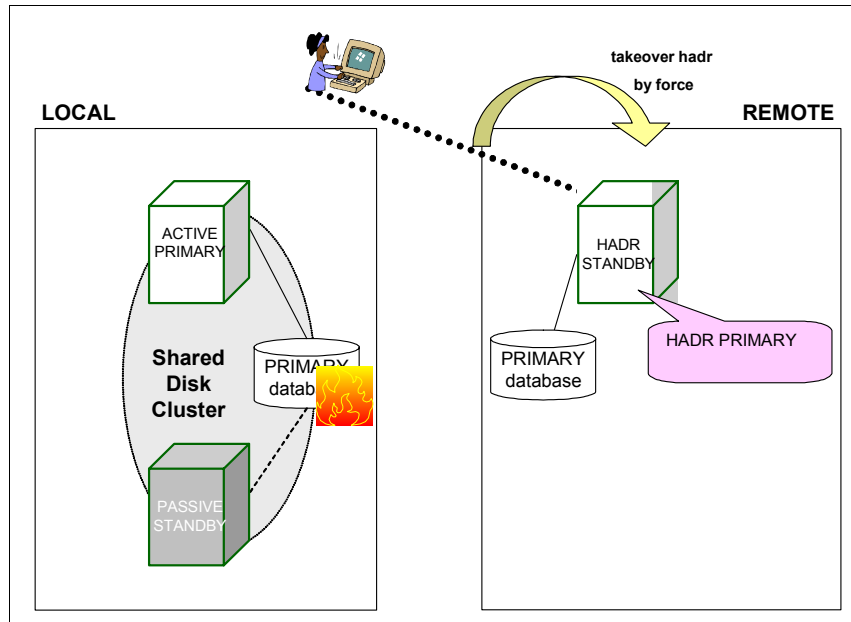


Figure 1-9 Crash or planned maintenance of shared storage device

► Cluster software maintenance:

When you are required to stop the cluster software to make some configuration changes and a failover test is required, you can fail over the services to the HADR standby database. Meanwhile, you can check the behaviors of the cluster software without any impact on the application services.

► Upgrading DB2 FixPak (Rotating upgrade):

You can upgrade DB2 FixPak with only an extremely short service downtime. Complete the following steps to upgrade DB2 FixPak:

- Update FixPak on HADR standby and re-establish the HADR connection.
- Fail over to HADR standby. The downtime for the application is just a few seconds of taking over the HADR role.

- c. Update DB2 FixPak in the shared disk cluster (active primary node and passive standby node).
- d. Switch the HADR role back to the local cluster.
- Using standby database as staging environment:

It is frequently necessary to test new applications or new database functions on a database, which has the amount of data (or data itself) similar to the product environment. After the standby database is detached from the HADR pair, it is available for a staging environment with complete data replicated from the product environment. With the hybrid system, you still have redundancy for local failure while you are utilizing HADR standby as the staging environment.

Staging environment is a test environment/infrastructure that mirrors the production and allows you to test new or changed application or settings without affecting the production system and the user, as shown in Figure 1-10.

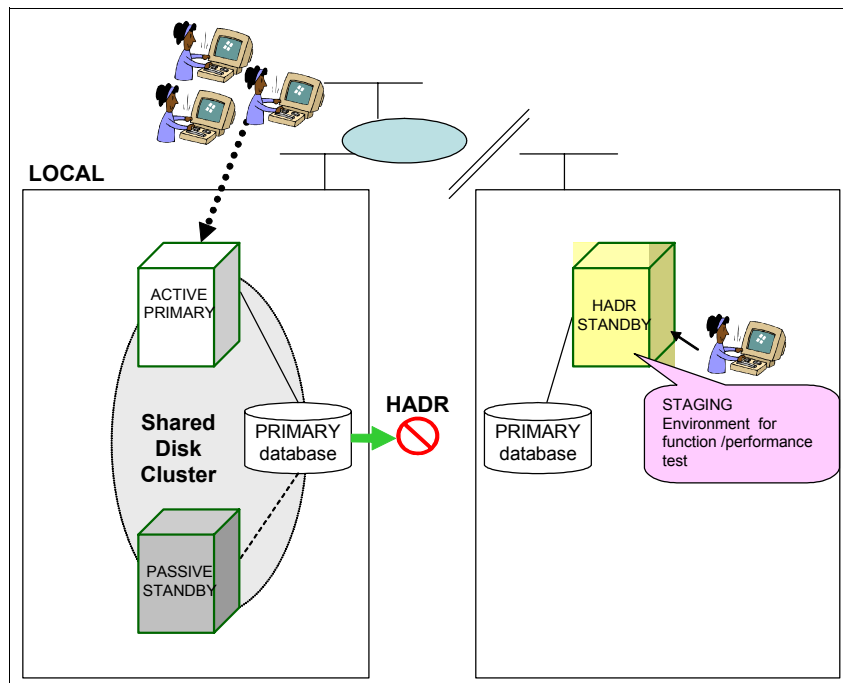


Figure 1-10 Use standby database as staging environment



# HADR introduction

Organizations today face a tough challenge in choosing an appropriate high availability solution that meets their business requirements and IT budgets.

Until recently, such solutions were based on proprietary systems, and involved significant investment in capital, time, and resources to assemble, integrate, test, manage, and support. This scenario has been changed dramatically with the introduction of software-based high availability solutions.

In this chapter we provide a general overview of an IBM software-based high availability solution: *DB2 High Availability Disaster Recovery (HADR)*.

We cover the following topics:

- ▶ High availability overview
- ▶ HADR overview
- ▶ Why HADR?
- ▶ HADR architecture
- ▶ Terminology

## 2.1 High availability overview

High availability (HA) is the term used to describe systems that run and are available to customers more or less all the time. For this to occur:

- ▶ Transactions must be processed efficiently, without appreciable performance degradations (or even loss of availability) during peak operating periods.
- ▶ Systems must be able to recover quickly when hardware or software failures occur, or when disaster strikes. DB2 has an advanced continuous checkpointing system and a parallel recovery capability to allow for extremely fast crash recovery.
- ▶ Software that powers the enterprise databases must be continuously running and available for transaction processing. To keep the database manager running, you must ensure that another database manager can take over if it fails. The concept of moving functionality from one server to another is variously known as *failover* or *takeover*, largely depending on the software vendor's terminology. Failover capability allows for the automatic transfer of workload from one system to another when there is a hardware failure.

### 2.1.1 Solutions for high availability

Now more than ever, customers are demanding a 24x7 operating environment. To implement this requirement, organizations must give high availability and disaster recovery a great deal of consideration.

Many customers use a hardware high availability (HA) approach, where all the application updates on one machine are synchronized to a backup machine. However, there are many advantages of using a software-based approach, chief among them being lower cost. IBM provides software-based HA solutions such as DB2 HADR, WebSphere Information Integrator Q Replication, SQL Replication, and HA clustering software — HACMP and Tivoli System Automation (TSA).

HADR is one among several replication solutions offered in the DB2 product family. WebSphere Information Integrator and the DB2 database system include SQL replication and Q replication solutions that can also be used, in some configurations, to provide high availability. These functions maintain logically consistent copies of database tables at multiple locations. In addition, they provide flexibility and complex functionality such as support for column and row filtering, data transformation, updates to any copy of a table, and can also be used in partitioned database environments.

## 2.1.2 Types of high availability

There are two types of high availability:

- ▶ Continuous availability:

Continuous availability requires that the database engine be available for processing SQL transactions without any downtime. Such availability is usually implemented in mission-critical business applications. For this to happen, total redundancy is required, which means you must have two systems that are fully independent of each other, both in terms of hardware and software.

- ▶ Failover availability:

Failover availability is differentiated from continuous availability by the fact that for some period of time, however small, the database engine or other critical server function is not available for transaction processing. The essential components for this type of solution are:

- Primary and standby systems
- Failure detection
- Data source movement

In the context of providing continuity for a database management system, the two systems (primary and standby), must each have access to copies of the database data, which clients can normally access from the primary. When the failure detection mechanism is triggered by a critical resource breaking on the primary, a failover takes place. In the failover process, the data source is moved from the primary to the standby system, and clients start accessing the data through that system. It should be pointed out that this process is seamless from the clients' perspective. Some solutions allow for the data source to be replicated, such that the primary and the standby data is stored separately. Others have the data in a single location, which is shared between a primary and a standby system, but only accessed by one system at any given time.

This book focuses on solutions providing failover availability, although to some extent, certain scalability solutions also provide a type of continuous availability.

## 2.2 HADR overview

DB2 High Availability Disaster Recovery (HADR) is a database replication feature that provides a high availability solution for both partial and complete site failures. A database server can fail from any number of factors: environmental (power/temperature), hardware, network connectivity, software, or human intervention.

Recovery from the loss of a database server in a standard installation can require a machine reboot and database crash recovery, which will interrupt the database services. By using DB2 HADR, you can safely minimize the downtime to only a few seconds. HADR protects against data loss by continually replicating data changes from a source database, called the primary, to a target database, called the standby. Furthermore, you can seamlessly redirect clients that were using the original primary database to the standby database (which becomes the new primary database) by using Automatic Client Reroute (ACR) and retry logic in the application. This seamless redirection is also possible using software solutions discussed in other chapters, which can manage IP address redirection.

Figure 2-1 illustrates the concept of HADR, where DB2's log buffer is used as a single source for all changes to the DB2 primary database server.

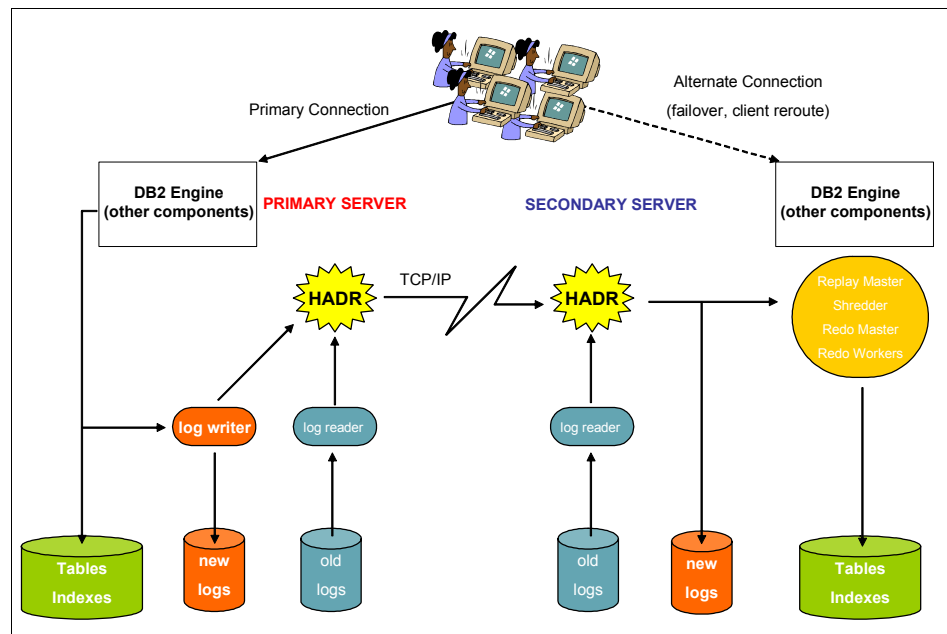


Figure 2-1 HADR overview

HADR transmits the log records from the primary database server to the standby server. The HADR standby replays all the log records to its copy of the database, keeping it synchronized with the primary database server. The standby server is in a continuous rollforward mode and is always in a state of near-readiness, so the takeover to the standby is extremely fast. Applications can only access the primary database and have no access to the standby database.



Using two dedicated TCP/IP communication ports and a heartbeat, the primary and the standby keep track of where they are processing currently, and the current state of replication; perhaps most importantly, whether currently the standby database is logically the same as the primary (known as *HADR Peer state*). In the Peer state, the HADR primary retrieves log data from its in-memory log buffer at the time the log pages from the buffer are flushed to the primary's disk and sends them to the standby through TCP/IP.

As mentioned, HADR communication between the primary and the standby is through TCP/IP which enables the database to be replicated to a remote geographical site. This allows the database to be recovered in case of a disaster at the primary database server site. The HADR solution thus provides both HA and disaster recovery ability. As you can appreciate, HADR, with the potential for no loss of committed transaction data, provides an incomparable improvement on conventional methods for DB2 disaster recovery, which often mean losses in terms of hours of committed transaction data.

If the primary system is not available, a takeover HADR by force operation will convert the standby system to be the new primary system. Once the maintenance or repair is done in the old primary system, you can bring the original primary DB2 up and return both DB2 servers to their primary and standby roles after reintegration of HADR.

In a DB2-ready environment, HADR is easy to set up and configured. The bottom line is that HADR is an efficient method of supporting high availability disaster recovery.

## Why HADR?

The reasons to choose HADR as a solution are as follows:

- ▶ Ultra-fast failover capability
- ▶ Negligible impact on performance
- ▶ Easy to set up and monitor
- ▶ Rolling upgrades without service interruption between non-major versions or changes requiring server recycling, and reduced change windows for other upgrades
- ▶ Transparent failover and failback for applications
- ▶ Easy integration with HA clustering software
- ▶ Dramatically improved disaster recovery compared to conventional methods

## 2.3 DB2 logical log architecture

In this section, we discuss the logical log architecture. This DB2 architectural background can be helpful for clearly comprehending the architecture behind HADR. We assume that the readers are familiar with the DB2 buffer pool architecture.

One of the main features that DB2 has to improve performance, and guarantee the security and robustness of your data, is the capability of managing transaction integrity.

In order to preserve the logical integrity of the database, individual SQL operations are grouped together as a transaction. All operations inside a transaction are guaranteed to be executed in an all-or-none fashion. If a failure occurs at any individual step, the transaction is aborted (rolled back) and all individual operations that have already been executed are undone. Transactions are also known as atomic, or logical units of work, in that while they can be constituted of smaller elements, they cannot be (and must not be) logically split up, either from the point of view of an application's specific business rules, or data integrity in general.

DB2's logical logging implements the Algorithm for Recovery and Isolation Exploiting Semantics (ARIES) recovery algorithm, which is a write-ahead logging (WAL) protocol to ensure atomic operations. In this protocol, every agent that modifies a page will also generate a record of the operation known as a *logical log record*. This logical log record is stored in a special section of the disk known as a *logical log file*. Transactions have special logical log records that identify the beginning and the end of a transaction. In the event of a crash, the logical log files are read and all records in them are replayed. All records belonging to the incomplete transactions are rolled back, bringing the database to a consistent state.

ARIES is described in further detail at the following Web site:

[http://www.almaden.ibm.com/u/mohan/ARIES\\_Impact.html](http://www.almaden.ibm.com/u/mohan/ARIES_Impact.html)

For performance reasons, the logical log records are not written directly to the disk, but to a buffer area known as the *logical log buffer*. To minimize the information lost, the logical log buffer is flushed every time an application commits a transaction.

For the ARIES protocol to work as designed, a condition must be met: no dirty pages in the buffer pool can be written to the disk (table space container) before the logical log buffer pages containing the logical log records that modified the buffer pool page are written to the logical log file.

Figure 2-2 depicts a sequence of events that happen in the server when an application requests an update to a page in a table space. The db2agent is a process (in Windows platforms it is a Windows thread) that is in charge of executing the work requested by the application.

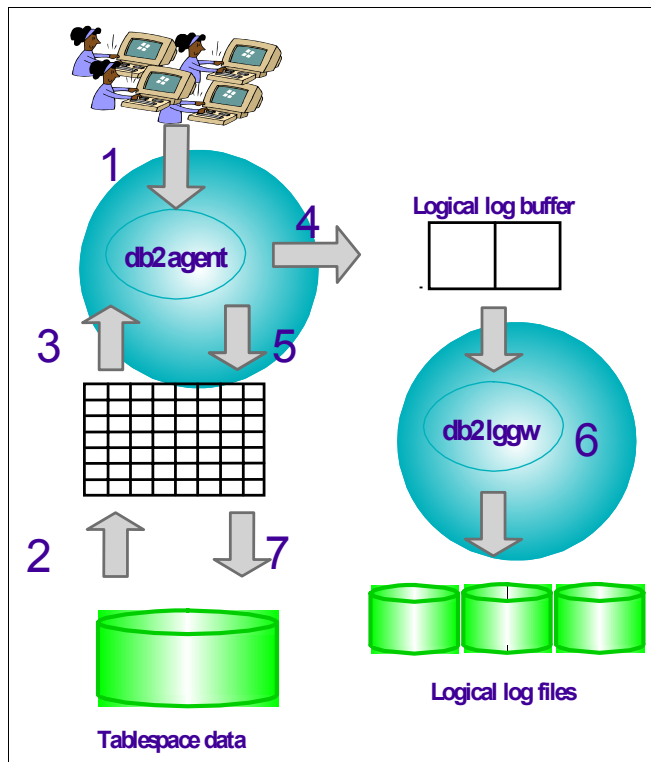


Figure 2-2 Application interaction with DB2 processes to update table space data

Here we describe each sequential event:

1. The application requests to update a row in a table.
2. The data is read (fetched) from the table space into a page in the buffer pool.
3. The db2agent receives the page from the buffer pool and modifies it.
4. The DB2 engine records what was changed in that page by writing a logical log record into the logical log buffer. This record can later be used to undo the operation in case of a rollback, or to replay the transaction in the event of a failure (rollforward).
5. The page is modified and placed again into the buffer pool. At this point it is not safe to write the page from the buffer pool back into the table space as the logical log record is still in the log buffer but is not written to the log file yet.

Because the log buffer only exists in the memory, a system crash can cause the logical log record to be lost forever. If this change were written to the table space before it were recorded in the log file, in the event of a DB2 crash, DB2 would not know about this transaction from the log files. DB2 would not be able to replicate that same data by means of a rollforward from the most recent backup image. Conversely, a crash after a successful write of log buffer data to the log file on the disk would enable DB2 rollforward recovery to replicate the data correctly in the table space container.

Note that because flushing the log buffer and the buffer pool are asynchronous operations accomplished by different engine dispatchable units (EDUs), db2agent is freed at this point to do any other work.

6. At some point an event triggers a logical log buffer to be written to the logical log files. This work is accomplished by an EDU called db2loggw. Events that might trigger the flush of the logical log buffer include:
  - An application issues a commit and commit count is equal or higher than the value specified in the database configuration parameter MINCOMMIT.
  - A page in the buffer pool that references a log record in this buffer must be written to the disk.
7. Once the logical record has been written to the logical log file, it is safe to flush the modified data page from the buffer pool into the table space.

Every operation executed by the database over a page in a table space generates a logical log record that is stored in a logical log file. Logging is made at the database level in DB2, so each database will have its own log files and separate logging processes.

The logical log buffer is written to the disk by the logger process db2loggw. There is another log EDU known as the *log reader* which has the role of reading data from the logical log files during, for example, a rollforward operation, or when a userexit application is invoked. The classical log reader used by DB2 version 8 is db2loggr.

With the introduction of HADR in DB2 V8.2, a new type of log reader architecture is created. This new architecture is implemented by an EDU known as *log file reader* (db2lfr). Presently, db2lfr and db2loggr coexist to manage different subsystems, but eventually, both of them will be migrated to the new db2lfr EDU.

## 2.4 HADR architecture

HADR technology is tightly coupled to the DB2 logging and recovery strategy. The main design goal of DB2 HADR is to quickly failover the failed system to a spare cluster node or to a disaster site in case of a failure on the primary server.

The basic architecture of HADR is very simple. It is based on the shipping of logical log records from the primary database to a remote instance that contains what is known as the “standby database”. Thus, in DB2 the HADR implementation allows for the database administrator (DBA) to decide which databases must be replicated.

The communication between the primary and the standby databases is accomplished by a communication protocol based in messages relayed over the TCP/IP layer.

After the appropriate start HADR commands have been entered on both the standby and primary databases, the HADR system actually starts once the primary can make a successful initial connection to an activated standby database. In any case the database can then only be deactivated by the **deactivate database** command.

In an HADR system, DB2 has two special EDUs to manipulate all HADR work. The one in the primary is db2hadrp. Its counterpart in the standby database is db2hadrs. They are in charge of sending logical logs from the primary to the standby, processing system messages, and receiving logical logs on the standby system.

At database activation in the primary, the db2hadrp is spawned; this will read the database configuration file and open a port to listen for a connection from the standby. The relevant HADR database configuration parameters are described in 6.1, “DB2 HADR configuration parameters” on page 138. During this time no client connections are allowed. The primary will wait for the number of seconds set in the configuration parameter HADR\_TIMEOUT for the standby connection, and will then abort the database activation if it is not able to successfully connect to an active standby database.

If the FORCE clause is used when issuing the START HADR command, the database will not wait for the standby to connect. The primary creates a listener and listens on the HADR port. The standby will initiate to connect to the primary. If the standby's connection attempt fails, it will retry connecting to the primary. But clients will be able to the primary database, and whenever the standby is ready, the HADR functionality will become operative.

The downside of using the FORCE clause is that the database will not respect the setting of the AUTORESTART parameter and in the event of a crash occurring, crash recovery will be performed irrespective of the current setting. Any other methods to start HADR, including ACTIVATE DATABASE and the first client connection, will respect AUTORESTART settings.

After the connection is established, the status and “health” of the HADR system is monitored by a mechanism known as *heartbeats*. This is basically to verify that

both the primary and the standby can see each other. Heartbeats are special short messages sent over the HADR connection from the primary to the standby, which are acknowledged and sent back by the standby. The heartbeat messages are spaced at known time intervals so each end can know how many heartbeats have been missed and take appropriate actions.

Upon successful connection of the standby database to the primary, the HADR system enters the *catchup phase*. During the catchup phase the size of the logical log gap from the standby to the primary is established and the primary starts sending all logical logs that are required for the standby to reach the same point as the primary.

The logical log reading is accomplished in the primary by the db2lfr EDU. This process relays logical log pages to the db2hadrp, that in turn will relay the pages over the TCP/IP layer for the db2hadrs in the standby to catch and relay to its logical recovery subsystem.

Once all the logs in the disk and the memory in the primary have been relayed to the standby, the HADR system enters the *Peer state*.

During the Peer state the db2lfr in the primary finishes its mission and passes the token to db2loggw. This db2loggw EDU that is in charge of writing the logical log buffers to the disk, has an extra task while in the HADR mode: immediately after the log buffer is flushed to the disk, db2loggw sends a message to db2hadrp with the location of the buffer and waits. db2hadrp then picks the log records from the log buffer directly and ships them to the standby server.

Upon completion of this log shipping, db2hadrp sends the log shipping status (either the log was shipped successfully or log shipping failed due to a TCP/IP error) to db2loggw so that it can continue processing log buffers.

Since db2loggw cannot proceed without the acknowledgment of db2hadrp, this is a crucial point for performance. Depending on the HADR mode, the speed to send the acknowledgement to the logger will be different. The completion of log shipping and sending status to db2loggw depends on the TCP/IP send status and the HADR synchronization mode being used. Until a log shipping status is received by db2loggw, the database log write activity cannot proceed on the primary system. If this happens, transactions on the primary system will be blocked until db2loggw receives such a status notification from db2hadrp.

Figure 2-3 shows the HADR primary database architecture details. The primary database ships the log records to the standby database server over the TCP/IP network (shown as the network zigzag on the right), when the primary does a log flush. The primary database server performs a log flush when the application issues a commit or when the DB2 log buffer (which is controlled by the database configuration parameter LOGBUFSZ) becomes full.

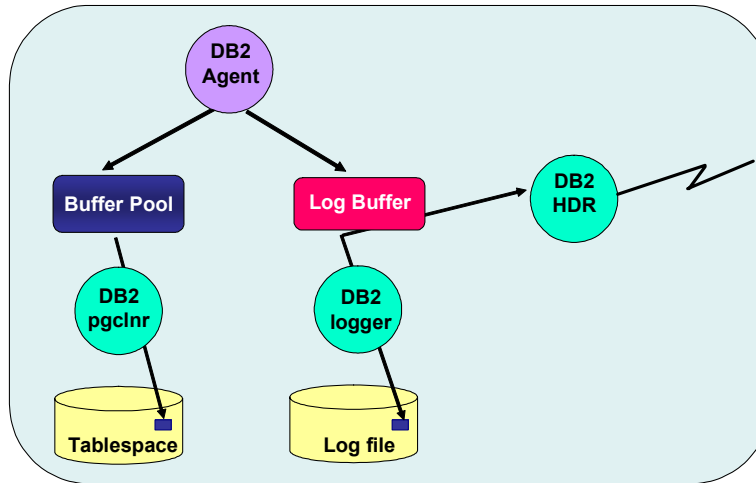


Figure 2-3 HADR primary database

Figure 2-4 shows the standby database architecture details. The standby receives logs through the TCP/IP link coming in from the left, into a HADR buffer (the size of which is controlled by the DB2 registry variable `DB2_HADR_BUF_SIZE`). On the HADR standby system, the standby EDU `db2hadrs` also manages log files (creation, recycling, and writing to log files). This EDU writes the log pages to its local disk.

Figure 2-4 HADR standby database

## Handling failure with the help of HADR

The mechanism of DB2 HADR for handling failure at the primary side, is through the `FORCE` clause in the HADR takeover command, as mentioned previously. Once this occurs, HADR only knows about the new primary system, and looks for the failed primary reintegrating or a new standby system that is initialized. Thus, the old primary system can no longer be validly used in its present state for clients to connect to, because of the potential loss in data integrity.

The process of repairing the old primary system must be done carefully so as to avoid an accidental start HADR by force. The situation where two databases are active in HADR primary mode, or one or more have had HADR stopped and are now in the *standard* mode, is known as *split brain*, which is described in 2.5, “Terminology” on page 36. Creating an environment which is safe for repair (no risk of accidental client connection) can be accomplished by disabling the remote client TCP/IP connectivity to the DB2 instance port number on that server, whether that is by means of IP address takeover, or firewall, or simply temporarily updating the database manager configuration port number for that instance to some other unused value.

After the failed original primary server is repaired, it can be re-established as a standby database if the two copies of the database can be made consistent. This process is discussed in 3.4.5, “Re-establishing HADR after failure” on page 86. After the original primary database is reintegrated into the HADR pair as the standby database, you can easily switch the roles of the databases to set the database on the repaired server to once again be the primary database, with an unforced HADR takeover command.

## HADR topology

The standard, and only currently supported topology for any one pair of databases is one primary and one standby database. This makes the most commonly seen layout as a pair of servers, one with a single DB2 instance with a single database acting as the primary, and the other with a single DB2 instance with a single database acting as standby.

However, this does not limit the ways in which multiple pairs of HADR databases can be implemented on multiple server nodes.

HADR replication takes place at a database level not at the instance level. So the standby server can have multiple databases from multiple primaries on it. Another configuration option is to implement two servers in an active/active mode cluster as shown in Figure 2-5. The advantage with this option is that it makes more efficient use of the available CPU cycles.



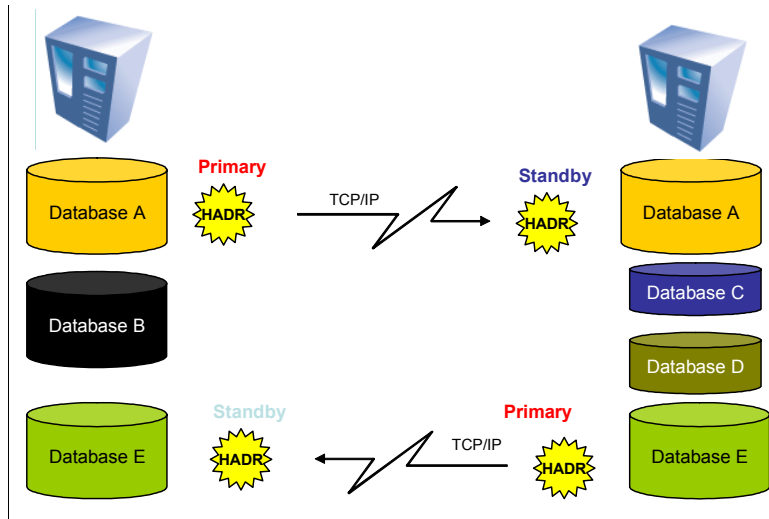


Figure 2-5 HADR takes place at database level not DB2 instance level

## HADR synchronization modes

With HADR, you can choose the level of protection you want from potential loss of data by specifying one of the three synchronization modes:

- ▶ **Synchronous:**  
Log write is considered successful only when logs have been written to the primary's log files and after an acknowledgement has also been received that it has also been written to the standby's log files. There can be no data loss in this mode if the HADR is in the Peer state.
- ▶ **Near-synchronous:**  
This is the default option. Log write is successful only when the primary's log buffer has been written to log files on the primary and an acknowledgement is received that log buffer has been received on standby. A double failure would then have to occur for data loss while in the Peer state; that is, if the primary goes down and the standby also fails before takeover is completed. In this case transactions in memory are lost. This is very unlikely, especially if the two servers are using independent power and operating environments.
- ▶ **Asynchronous:**  
Log write is successful when logs have been written to the disk on the primary and log data has been delivered to the TCP/IP layer. Data loss is more likely in this mode. A failure of the primary, network, or the standby can cause log data loss. If the primary becomes available again, then standby can still synchronize successfully. If takeover is needed due to a primary failure while log data remains outstanding, these will never make it to the standby database, and transactions will be lost.

Figure 2-6 illustrates these HADR synchronization modes, and the point at which DB2 considers itself able to commit a transaction while in the PEER state.

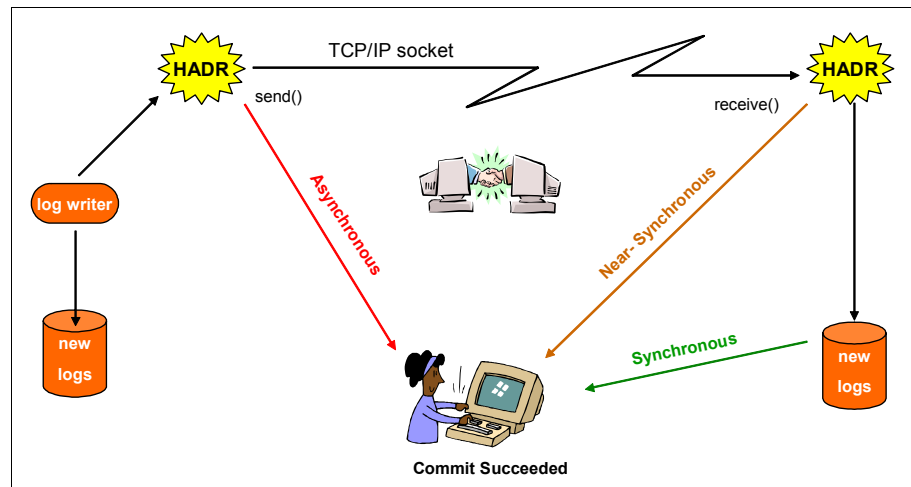


Figure 2-6 Synchronization modes

## 2.5 Terminology

Here we list and describe some of the more common terms used specifically with HADR. Most of these terms can be found in various DB2 manuals or the DB2 Information Center, where you can search and see them being used in context:

► Catchup phase:

HADR initialization always starts in the catchup phase, in which the standby tries to *catch up* to in-memory logs on the primary by replaying logs that have been written to the disk or the archive. During catchup, the standby can retrieve log files locally, or remotely from the primary through the HADR network connection.

► Failover:

This refers to changing the status of the standby in an HADR pair to become the primary, with full DB2 functionality, due to the original primary failing. The original primary can be brought up subsequently in the role of standby. Care must be taken to ensure that the original primary is truly non-functional at the time of failover. If both databases are functioning as primary, there will be conflict in the data update that HADR cannot resolve. The result is two incorrect versions of the database, which might be impossible to reconcile.

► **Failback:**

After a failover has occurred, which made the original standby into a primary, failback is the act of reverting this database back to a standby and bringing the original primary back as a primary once again. That means, switching the roles of the primary and the standby, while both are alive and healthy.

Failback is not a mandatory operation. A customer can choose to leave the databases in their reversed roles until another failover is necessary due to the failure of the new primary. This is especially likely in cases where the HA goal is ultra-fast failover, as failback can be viewed as a needless additional interruption in service.

Customers who choose this option must ensure that all DB2-related objects and processes which HADR does not replicate, are the same on both the servers; especially things such as batch schedule tasks, which must be deactivated while the database is in the standby mode, and activated while it is in the primary mode.

► **Failure:**

Failure is an event where any database service prerequisite component (DB2, operating system, server machine or network) is no longer able to provide the service it is supposed to. HADR will maintain data availability in the event of failure of any single component. If the failure affects only the standby system or the communication between the primary and the standby, data remains fully available on the primary without any disruption or required user action.

If the failure prevents the primary itself from providing DB2 functionality, the user can take the primary DB2 instance, the server, or both completely down (if it is not already) and initiate failover, which will rapidly change the standby system into a primary, making the data available again after only a brief outage.

Some failures can be detected and reported by HADR and DB2 to assist the user in deciding whether failover is needed or whether certain database components simply require attention. The HADR database processes will report a failure when they are unable to communicate. The DB2 database processes will report a failure when they detect a fault. These are the only DB2 database components that will report failures as provided by the DB2 software.

► Out of band:

In this book, out of band refers to operations that can be performed on the primary database, and which must be replicated at the standby, but cannot be repeated solely based on log records. Such operations are non-logged or not completely logged at the primary. In order to replicate such an operation, its results (a copy of the new or modified object, for example) must be passed to the standby by some other means. This “other means” is considered to be *out of band* with respect to the log stream. Note that this use of the term is different from the meaning of *out of band* in TCP/IP. Thus, it is preferable to refer to the specific operations or to *non-logged* or *not completely logged* operations to avoid any confusion.

► Outage period:

The outage period is the amount of time it takes to failover DB2 functionality from a broken DB2 primary database to the standby. This period of time starts from the point at which failover is initiated to the point that DB2 functionality is restored. To a client connecting to the DB2 database, it will be seen as a connection failure, at which point retry logic should be working to ensure that the transaction is not lost.

► Peer state:

After the standby catches up with in-memory logs on the primary, HADR enters the Peer state, in which the primary ships the log page to the standby whenever it flushes a log page to the disk. The log pages are replayed on the standby as they arrive. The pages are also written to local log files on the standby so that the primary and the standby have identical log file sequences. Transaction commits on the primary will wait for acknowledgment messages from the standby or at least for a successful return of log send calls, depending on the user-specified log shipping mode level.

► Primary (database):

This is the principal (master) copy of the database. Applications apply updates to the primary database and those updates are propagated to the standby server via log shipping.

► Primary reintegration:

In some cases, the old primary database can rejoin an HADR pair after a failover. Because the old standby is now the primary, the old primary can only join as a standby. This is called primary reintegration. After primary reintegration, a failback can be performed optionally.

► Standby (database):

This is a copy of the primary database. It is not updated directly by the application. All updates occur by rolling forward log data generated on the primary database.

► Split brain:

This refers to the condition of having both databases in an HADR pair acting as primaries simultaneously. Such a situation leads to the databases becoming inconsistent with each other and is thus to be avoided at all costs. HADR will never automatically make state transitions that result in two primaries, and when the pair can communicate with each other (that is, the network connection between them is operating properly), they will also prevent a user from creating this situation in an unforced takeover. HADR will also not allow an inactive primary database to be activated without a successful connection to a standby within the HADR\_TIMEOUT period.

However, split brain is still possible in a situation where the user instructs the current standby to become a primary while the communications link between the pair is down, if either the current primary is still active or if it is brought back to the non-HADR, or standard mode. There is nothing HADR can do to prevent this, because in order for HADR to achieve its main HA goals, there must be some way to change the standby into a primary in situations where the original primary or the network is inoperative (for example, the building housing the original primary burns down, but the standby is safe in another building), so an HADR standby must obey the orders of the user to become a primary if it is no longer able to talk to the original primary.

► Standard (database):

In the context of HADR, *standard* means a normally operating non-HADR database. That is, a database not using the HADR feature, and therefore not operating in either the primary or the standby mode.

► Synchronous mode:

In the Peer state, the primary will not consider a transaction as committed until it gets an acknowledgment message from the standby confirming that the relevant log data has been received and written to the disk on the standby. Therefore, if a transaction is committed on the primary, it is guaranteed to be persistently stored in the standby's log file. Even if the standby crashes before it is able to replay the log, it can still replay it from its own log file when it restarts. There will be no transaction loss in a failover as long as the primary was in Peer state at the time of the failure.

► Near-synchronous mode:

In the Peer state, the primary will not consider a transaction as committed until it gets an acknowledgment message from the standby confirming that the relevant log data has been received and written to the main-memory of the standby. There is a very small chance that some transaction data would be lost should a failover occur.

► Asynchronous mode:

In the Peer state, the primary will not consider a transaction as committed until it successfully submits the relevant log data to the network. The primary does not wait for any acknowledgment message that the log data was received. There is a high chance that some transaction data would be lost should a failover occur.

► Takeover:

Takeover is the act of the HADR standby taking over control of the database from the old primary server and becoming the new HADR primary. Takeover is always initiated from the standby. If the primary can be reached over the network as in an unforced takeover, the standby will ask it to switch to standby, performing cooperative role switching. Otherwise, the standby takes action unilaterally (with the risk of dual primary/split brain). Failover is a unilateral takeover. Failback is a form of cooperative role switching performed after first reintegrating the repaired server as a standby. Note that HADR *takeover* can be performed outside the context of failover and failback. A user might want to switch HADR roles for rolling upgrade, which does not involve a failure at all.

► Engine dispatchable unit (EDU):

The engine dispatchable unit (EDU) can be implemented either as a process or a thread, depending on the platform architecture.

► Primary reintegration:

In some cases, the old primary database can rejoin an HADR pair after a failover. Because the old standby is now the primary, the old primary can only join as a standby. This is one form of HADR reintegration. After reintegration, a failback can optionally be performed. See 3.4.5, “Re-establishing HADR after failure” on page 86 for steps involved in reintegration.



## HADR setup

In this chapter we describe the steps to set up a High Availability Disaster Recovery (HADR) environment using both command line and Graphical User Interface (GUI). We cover the basic operations, including START HADR, STOP HADR, and TAKEOVER HADR commands. We also provide troubleshooting tips.

## 3.1 Requirements for setting up HADR

Before you start your setup of HADR, you must first know the requirements for setup. As part of this book, we describe the basic outline of what you need to get started (for more details, refer to *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228). Along with the requirements, we have also added our recommendations.

### Requirements

Setting up an HADR includes the following requirements:

- ▶ HADR is a DB2 optional licensing feature. One license is included in the Enterprise Server Edition (ESE).
- ▶ The operating system on the primary and standby databases should be the same version, including the patches. For a short time, during upgrades, they can be different.
- ▶ A TCP/IP interface must be available between the HADR primary and the standby.
- ▶ The DB2 version and level must be the same on both the primary and the standby systems. For a short time, during upgrades, they can be different.
- ▶ The DB2 software for both the primary and the standby database must have the same bit size (32-bit or 64-bit).
- ▶ Buffer pool sizes on the primary and the standby should to be the same.
- ▶ The primary and standby databases must have the same database name. This means that they must be in different instances.
- ▶ Table spaces must be identical on the primary and standby databases including:
  - Table space type
  - Table space size
  - Container path
  - Container size
  - Container file type
- ▶ The amount of space allocated for log files should also be the same on both the primary and standby databases.

### Highly recommended

These are some of the highly recommended parameters to set up an HADR:

- ▶ Use identical host computers for the HADR primary and standby databases. That is, they should be from the same vendor and have the same architecture.



- ▶ Use a high-speed, high-capacity network for the TCP/IP interface.
- ▶ Ensure that the primary and standby databases have equal amounts of memory.
- ▶ Have identical database (DB) and database management (DBM) configuration parameters.

## 3.2 Setup and configuration

Here we perform the steps to set up HADR on an existing installation of DB2. Our Lab example uses 32-bit DB2 9 for SuSE 10 Linux on Intel® (kernel 2.6). However, these steps are applicable to all candidate environments.

### 3.2.1 Preparing the environment

In order to prepare a real world pre-change window environment to set up HADR, there are a few things you can plan for, which will allow you to be ready well beforehand. Unless you are currently still using circular logging, the only downtime required for setting up HADR is for updating database configuration parameters. This can be avoided by schedule the work in a regular maintenance window. There should be no pressure to minimize your change window, other than a recommendation to avoid periods of heavy logging if you want to avoid a long wait for the standby to catch up before reaching the Peer state.

If you are still using circular logging in your real world environment, you might want to set aside a separate change window before commencing HADR setup to plan for the changeover to linear logging, and all the associated considerations for log archival, overflow, housekeeping, log file size, and retry on failure. If you are testing HADR setup on a sandbox environment, this is only a minor consideration, and no separate planning is required.

As basic components for this Lab, you require two DB2 instances of the same FixPak level. You can expect to see something like the following text in a pop-up message if your instances do not match:

```
The selected standby system is not at the same DB2 version as the
primary system. HADR requires identical DB2 version on the primary and
standby systems. MYSERVER - DB2 - DB2Version(8,2,6). LEAD - DB2_LEAD
(db2inst1) - DB2Version(9,1,0)
```

HADR will work even if both instances are located on the same server, but a real-world environment will have separate servers, as we have used in our example.

Ascertain whether you will use the GUI HADR Setup wizard or the DB2 *Command Line Processor* (CLP) to configure HADR. The GUI wizard is perfect for introducing people who are new to the concept of HADR, but it has all the complexity and high network bandwidth overheads of a GUI. Ensure that you can get a fast and reliable display of the DB2 GUI interface. This is even more important if you are intending to use an X11 emulator for remote setup of a Linux/UNIX system.

The DB2 CLP environment is suitable for experienced administrators or for repeated configurations on multiple HADR server-pairs. It is also suitable for environments where it is impractical to get a GUI interface working, whether locally or remotely, such as AIX with telnet-only access, or systems with no Java™, or systems with limited administrative network bandwidth.

If you work in a high-security environment where people performing DB2 installs and maintenance are not given `sesu/sudo root` or Windows Administrator authority other than in a special change window, take the necessary steps to request that authority.

If you are running HADR setup from the GUI wizard as the DB2 instance ID for example, this specifically requires that the DB2 instance ID has the authority to update the `/etc/services` file (`%Systemroot%\system32\drivers\etc\services` on Windows), unless these ports have already been registered. You might also have to manually update the `/etc/hosts` file if you do not have a reliable Domain Name System (DNS) in place, but still want to use host names rather than IP addresses.

The two servers you would use in a real world implementation must be connected by a very reliable TCP/IP connection. They must preferably have no firewall between them, or at least with port 523 for DB2 Administration Server, the main port for each DB2 instance, and the two ports you will set aside for HADR communication must be left open.

Depending on your environment, and your choice of HADR SYNCMODE, you might be working with physically separate servers:

- ▶ Over an intranet WAN (recommend ASYNC or NEARSYNC; more for disaster recovery functionality as opposed to a strictly high availability implementation)
- ▶ On a LAN (recommend NEARSYNC)
- ▶ On servers sitting physically next to each other in the same rack (recommend SYNC)

Our Lab example uses the latter configuration.

If your environment is not tolerant of performing frequent backups, even if they are online, then take whatever steps are necessary to get a window in which to perform one as a starting point for the HADR change window.

If you want to reduce the number of steps performed inside the HADR GUI wizard as much as possible, in addition to changing from circular to linear/archival logging, you can catalog entries for the remote system on both the servers beforehand:

- ▶ ADMIN TCPIP NODE for the DB2 Admin Server
- ▶ TCPIP NODE for the DB2 instance
- ▶ DATABASE alias for the DB2 database

Example 3-1 and Example 3-2 show the commands to catalog the database server and database.

*Example 3-1 Commands to catalog remote entries on primary server (LEAD)*

---

```
db2 catalog admin tcpip node POLONIUM remote x.x.x.x remote_instance
db2inst1 system POLONIUM
db2 catalog tcpip node DB2_POL remote x.x.x.x server 50001
remote_instance db2inst1 system POLONIUM
db2 catalog database SAMPLE as SAM_POL at node DB2_POL
```

---

*Example 3-2 Commands to catalog remote entries on standby server (POLONIUM)*

---

```
db2 catalog admin tcpip node LEAD remote x.x.x.x remote_instance
db2inst1 system LEAD
db2 catalog tcpip node DB2_LEAD remote x.x.x.x server 50001
remote_instance db2inst1 system LEAD
db2 catalog database SAMPLE as SAM_LEAD at node DB2_LEAD
```

---

One final (and optional) consideration concerns the host and port entries for each server. The HADR GUI wizard will attempt to add two port entries for HADR into /etc/services, but you can arrange to have them added beforehand, especially if your local operating system security administrator will not give the DB2 instance ID enough authority to do so. Remember that if you do this, you will be required to override the default entries that appear in the wizard, as it tries to auto-increment the port numbers to something that does not already exist in the /etc/services file.

For our Lab example, we use HADR ports 55001 for LEAD, and 55002 for POLONIUM. You can also arrange to have the unqualified remote host name registered in /etc/hosts file on both the servers, so that you do not have to rely on IP addresses. Because they are not generally dynamic values, it is acceptable in the wizard to enter the physical port number for the local and remote DB2 instances rather than the service name alias (port 50001 for both instances in our Lab). For administrative and documentation reasons, it is still important to register the port numbers for the local services in each server's /etc/services file.

Having completed this, you are now ready to perform an HADR setup.

### 3.2.2 Configuration using the Setup HADR wizard

Figure 3-1 illustrates the basic intended high-level architecture of our Lab environment.

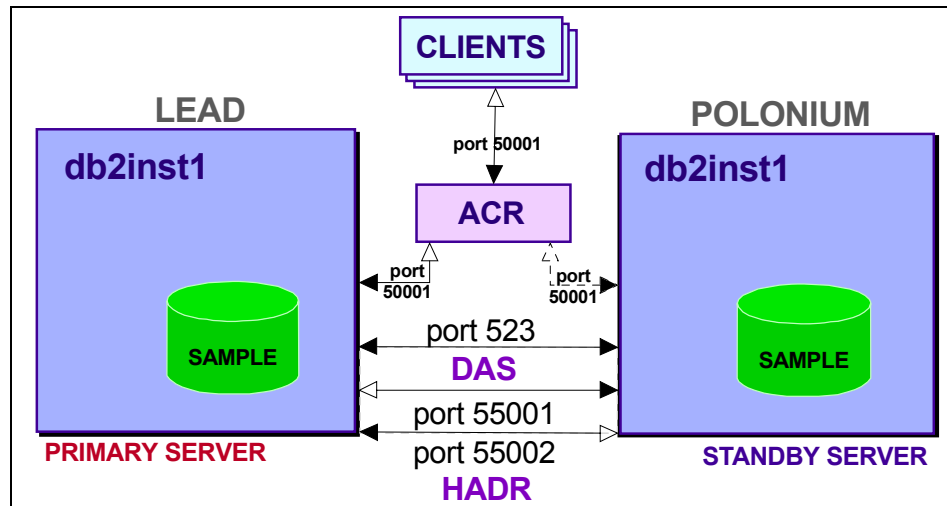


Figure 3-1 Lab environment high-level overview

In Figure 3-1, the two example instances are both called DB2INST1 on server host names LEAD and POLONIUM, in a non-clustered configuration, with basic TCP/IP ethernet network adapter connectivity. LEAD will initially be set up as the HADR primary, and POLONIUM will initially be the HADR standby.

The ACR box refers to *Automatic Client Reroute*, which is effectively a software layer sitting on top of HADR, between external clients, and each DB2 instance. It is there to route the Client to whichever DB2 instance is currently set as the primary. For more details about ACR, refer to Chapter 5, “Automatic client reroute” on page 115.

In our Lab, clients can communicate with the DB2 instance through port 50001, and DB2 HADR communicates between the servers through ports 55001 and 55002. The DB2 Admin Server (DAS) GUI interface communicates through port 523.

The database being set up in HADR configuration is called SAMPLE; we will eventually give this database an alias on each respectively remote server, so that it is accessible from the GUI Control Center Manage HADR dialog box, and for general remote takeover, stop, and start commands.

In the Lab, we start out with only one SAMPLE database, sitting in the DB2INST1 instance on one of the servers (LEAD).

We finish with SAMPLE acting as HADR primary on the DB2INST1 instance on LEAD, and as HADR standby in the DB2INST1 instance on POLONIUM.

In the step descriptions that follow, the term *remote* is used in the context of the server that we eventually configure the *standby* database on, and likewise, the term *local* is used interchangeably with *primary*. This is simply because we are running the wizard locally on the server that will become the primary. Initially, we do not have a primary or a standby, just the local server where the setup GUI dialog boxes appear (LEAD for our example) and the remote server that will get a database restored on it (POLONIUM).

After configuration is complete, the primary and standby roles can be swapped at will, so these are temporary descriptions at best. To help with establishing a more permanent frame of reference, we have included our Lab example server names (LEAD and POLONIUM) in parentheses when listing a given server, so you can substitute your own server names for each situation.

## Beginning HADR configuration

For beginners, the most user-friendly way to achieve a working HADR environment is to use the GUI wizard.

1. Start the DB2 Control Center (db2cc), and expand the **All Databases** tree on the left pane to show your databases.
2. As shown in Figure 3-2, right-click the database you wish to set up, and click **High Availability Disaster Recovery** → **Set Up ...**

**Note:** You might be prompted to enter a user ID and password here. This should be any user ID that has DB2 SYSADM authority on the local/primary instance, DB2INST1 in our example.

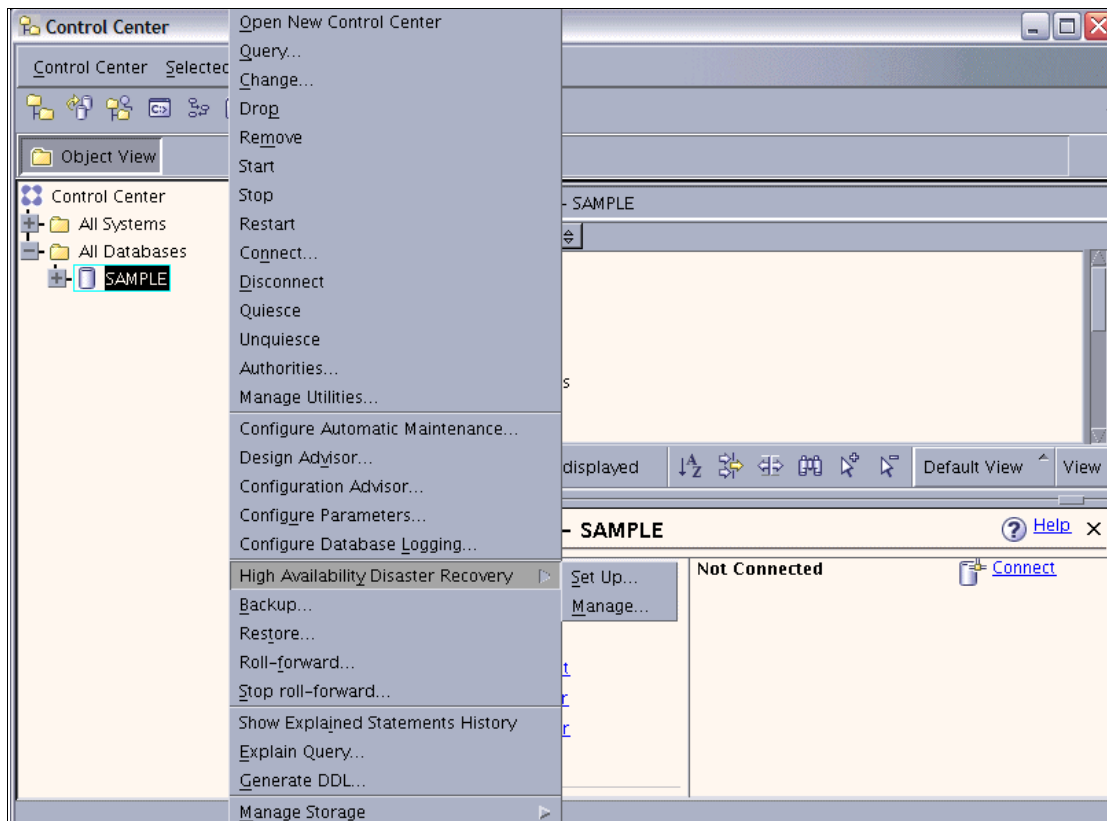


Figure 3-2 Starting HADR Setup wizard

3. The next screen is an introduction, giving you notes to keep in mind. Click **Next** after reading them briefly.

4. *Confirm the primary database selection* (see Figure 3-3). This points out that the database must use linear (non-circular) logging, and must not be in a partitioned environment.

If the database is still using circular logging, click **Configure...**, and follow the subsequent wizard steps to configure linear/archival logging.

If the database is already in the linear or archive logging mode, you can skip the next several sub-steps (step a through step k) and go straight to step 5.

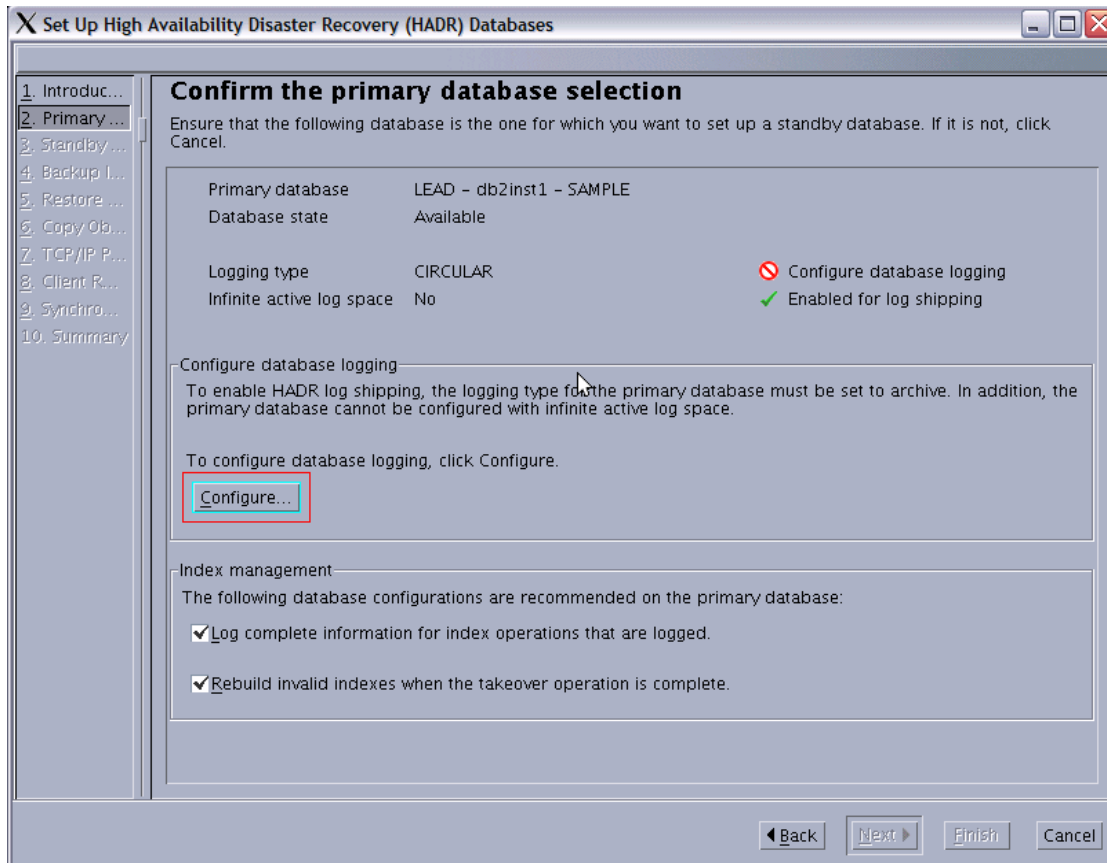


Figure 3-3 Confirming the database to use as primary, and configure it to not use circular logging

**Note:** Ignore the “inconsistent” state reference. Databases that are active are usually inconsistent. It just means that not everything has been written from log buffers to disk yet, or a pending state, or uncommitted activity.

- a. If the database is in the circular logging mode, the first dialog box that opens after you click **Configure** is *Choose your logging type*. Select **Archive logging**. Click **Next**.
  - b. *Choose how you would like to handle your archived logs*: We would generally select **Use DB2 to automatically archive the log files**, and then specify a location such as `/usr/db2/archlog` (\*nix) or `C:\db2archlog` (Windows). Click **Next**.
  - c. *Choose the number and size of your log files*: We accept defaults here, but refer you to *DB2 9.1 Administration Guide: Planning*, SC10-4223, Chapter 5 for recommendations based on your requirements. Click **Next**.
  - d. *Specify the location of your log files*: We would specify a location such as `/usr/db2/actlog` (\*nix) or `C:\db2actlog` (Windows). Click **Next**.
  - e. *Specify where to store your backup image*: This is required as part of the conversion to archive/linear logging, as the database is placed in backup pending after changing critical database configuration parameters, and logging recommences in the new linear format from that point in time. We would specify a location such as `/usr/db2/backup` (\*nix) or `C:\db2backup` (Windows). Click **Next**.
  - f. *Specify options for the backup*: Accept the defaults, compress the image if you want to save time transferring the file to the standby server later on. Click **Next**.
  - g. *Review the actions that will take place when you click Finish*: This is just a summary of the steps that will occur. You can see the actual commands by clicking **Show Command**. Click **Finish** or click **Back** to fix any incorrect settings as necessary.
  - h. A pop-up message is displayed, warning you that the database will be taken offline for a full backup. Click **OK**.
  - i. The DB2 Progress pop-up goes through the motions here. Wait for it to disappear.
  - j. The DB2 Message pop-up gives you a result summary. This should be successful, Click **Close**.
  - k. Now we are finally back at the dialog box we left in step 4, and can proceed. Click **Next**.
5. *Identify a standby database*: Once your database logging is in a state ready to start HADR configuration, and has been deactivated, you can proceed with the next step, shown in Figure 3-4. This dialog box is displayed irrespective of whether you already have a database sitting on another server. All you need at this stage is to have a DB2 instance running there.



The wizard knows whether the database has been successfully configured behind the scenes, and will refresh the status to allow you to proceed. You might be asked for the DB2 Administration Server user ID and password in order for DB2 to connect to the standby server, copy backup files, and so on.

**Set Up High Availability Disaster Recovery (HADR) Databases**

1. Introduc...  
2. Primary...  
3. Standby ...  
4. Backup I...  
5. Restore ...  
6. CopyOb...  
7. TCP/IP P...  
8. Client R...  
9. Synchro...  
10. Summary

### Identify a standby database

Select the standby system and instance. If the standby system and instance are not currently cataloged, click Add to catalog them.

System name: POLONIUM [Add...]  
Instance name: DB2\_POL (db2inst1) [Add...]

Standby database initialization options—

☒ Use a backup image of the primary database to initialize a standby database  
You can select from a list of existing backup images or create a new backup image. The following pages will help you to initialize the standby database using the DB2 [backup](#) and [restore](#) utilities.

☐ Use another existing database as the standby database  
Enter a database alias name to catalog the standby database.  
Database name: SAMPLE  
Database alias name: [Add Database...]

☐ Use suspended I/O and online split mirror to initialize a standby database  
To create a split mirror, consult the storage vendor documentation applicable for your device. Exit from this wizard to perform the split mirror procedure. Once you have created the mirror database, return to this wizard to configure the new database as the standby database. For more information, see [Suspended I/O and online split mirror](#).

[Back] [Next] [Finish] [Cancel]

Figure 3-4 Choosing a database to use as standby

- a. If you have not already done so, the wizard will prompt you to catalog a standby server IP/host name, Instance node name, and to either specify a backup image off the primary server, or an existing remote database. In order to have discovery working, the DB2 Administration Server must be running on the standby server (in our example, on POLONIUM).

Figure 3-5 shows the second dialog box in this series asking for an *Instance node name* to use for the remote instance, host address, and port number. The first dialog box is very similar, asking for a database alias to use for the remote database.

**Add Instance**

POLONIUM

Instance name:

Instance node name:

Operating system:

Protocol:

Protocol information

Host name:

Service name:

Port number:

☐ Enable TCP/IP SOCKS security

Figure 3-5 Catalog an Instance node name for the remote server's DB2 instance

As a brief explanation of these sub-steps, when you click **Add...** against *System name* and later against *Instance name* in Figure 3-4 on page 51, you can find yourself adding the host name/IP address for the remote *System* and assigning a node name, and then adding the host name/IP address for the remote *Instance* and being asked again to assign a node name. If you specify the same node name, the wizard reports that a node with that name already exists, and fails with an SQL1018N error.

Different node names must be specified for the System (the name you want the Control Center to display for the server/host name), and for the Instance (the name you want the Control Center to display for the instance).

In the *Add System* dialog box, the three mandatory fields for details of the remote server (System, Host, and Node name) are described as follows:

- **System name:** If you do not have a working DNS and you will use the IP address of the server in the next field, put the host name value here. By default, entering a value in this field first will fill out all the three fields with the same value, which is usually acceptable in a DNS environment with good naming standards.
- **Host name:** Either the IP address, the DNS, or the host name of the server registered in the `/etc/hosts` file.
- **Node name:** This can be any user-assigned value. Within DB2, it is a nickname, which represents the server, as opposed to the *Instance node name*, which represents the instance. In Windows, this is meant to refer to the NetBIOS Workstation name (dbm cfg parm: NNAME). There should be a hover/pop-up with this descriptive instruction if you hold the cursor on the node field. Practically speaking, it represents an ADMIN TCPIP NODE, which you can get a current list of with the command:

```
db2 list admin node directory
```

The remote server's dbm cfg might not even have a NetBIOS name defined, so you can just use any appropriate name up to eight characters.

The fact that you are required to catalog the *Admin TCPIP Node* partially explains why you must have the DB2 Admin Server running at both ends in order to set up and manage HADR using the GUI interfaces and wizards.

When you are finished with the Add System dialog box, click **OK**. You might be prompted at this point by the DB2 Admin Server to enter a SYSADM authority user ID and password for connectivity to that new system entry. If you receive a communication failure after correctly entering a valid user ID and password in this dialog box, it could be related to the firewall for port 523, but it is best to check through all the possibilities described in the troubleshooting guide presented in 3.4, "Troubleshooting" on page 80.

If you receive the SQL1018N error message due to a duplicate node name, and you want to remove the incorrect entry and use values in your Add System dialog box without cancelling the Setup HADR wizard, you can open a separate Command Window, and enter the command:

```
db2 uncatalog admin node <nodename>
```

In the *Add Instance* dialog box, the *Instance node name* is the nickname you create for the remote instance, cataloged for use on your local system (this is what gets displayed in the Control Center).

Remember, the System node name and the Instance node name are not meant to be the same. Behind the scenes, the System node definition is cataloging an ADMIN TCPIP NODE entry; the Instance node definition is cataloging a regular TCPIP NODE entry. That is why they are two separate layers in the DB2 Control Center tree structure.

One tip to avoid conflicts, while inside the wizard, is to add the system and instance definitions beforehand. This can be done with the Control Center or the *Configuration Assistant*. However, if discovery of the remote objects is not working for whatever reason, you will have to catalog the remote server, node, and database alias manually; see Example 3-1 and Example 3-2 on page 45.

If you cataloged the ADMIN TCPIP node with a system name (for example, Figure 3-5 on page 52 shows the system name value of `polonium.itsosj.sanjose.ibm.com`), you might have to insert the TCP/IP address and host name entry for the remote host/server (POLONIUM) in your local `/etc/hosts` file. For Windows, this address and host name would have to go in the file: `%Systemdir%\system32\drivers\etc\hosts`). Otherwise, you can get TCP/IP errors such as those shown in Figure 3-3.

---

*Example 3-3 No local entry in /etc/hosts for the remote hostname*

---

```
SQL22212N  A DB2 Administration Server communication error has
been detected.  Client system: "x.x.x.x".  Server system
"x.x.x.x.x".
```

---

This is because the DB2 Admin Server on the remote system might have the host/server name down in text rather than a literal IP address, and that is what DB2 on the local host will try to resolve, even when you have specified literal IP addresses for your remote System and Host names. Note that you can receive the SQL22212N error for other more generic reasons, listed in the pop-up message at the time you receive the error message.

For the fields specifying the service name and the port number, we have used port 50001 for both DB2INST1 on the primary and DB2INST1 on the standby — that is, the primary DB2 port for the DB2 service for the instance on each server; this is the port with which clients connect to DB2.

HADR itself uses two different ports, one for each server. We use 55001 and 55002; these ports are for log transmission and HADR administrative traffic.

After having avoided or corrected the foregoing communication issues, a prompt should appear for a user ID and password to connect to the remote instance; use any SYSADM user ID from the remote instance (POLONIUM). The pop-up message will show an error code SQL22201N with reason code “1”, but that is expected, as you have not entered the DB2 user ID or password yet.

- b. The *Standby database initialization options* section in Figure 3-4 on page 51 gives you three options to initialize the database on the standby system. The first one (*use a backup image of the primary database to initialize a standby database*) should give a simple and successful HADR setup. That is what we use.
6. *Specify a backup image of the primary database* (see Figure 3-6): If you have just set up your database for linear logging in this wizard, you must have created an offline backup, which is perfect for use here in HADR configuration.

If you were already using linear/archival logging, this is the point at which you choose **Select a backup image from the list provided** and choose the latest backup, or if the backups listed are not recent, run an online backup now with **Backup the primary database**.

The wizard is able to use DB2 ports to file transfer the backup on the local system across to the remote system. If the backup image which you will restore from is not listed here — for example, if the backup image is being transferred to the remote server over ftp or through filesharing — then click **Enter the backup image information** and specify the subdirectory (DB2 Admin Server is capable of navigating the file system of the remote server and also the local server), and the date and time for the backup image you will be using, as found in the backup file name.

If you use this method, click **Next**, and you are presented with the restore dialog as shown in step 7. This will appear similar to Figure 3-7 on page 57 but without having to copy the backup image across if you specified the location as being on the remote server.

In our example, we choose **Select a backup image from the list provided** as shown in Figure 3-6 and select the latest backup.

Click **Next**.

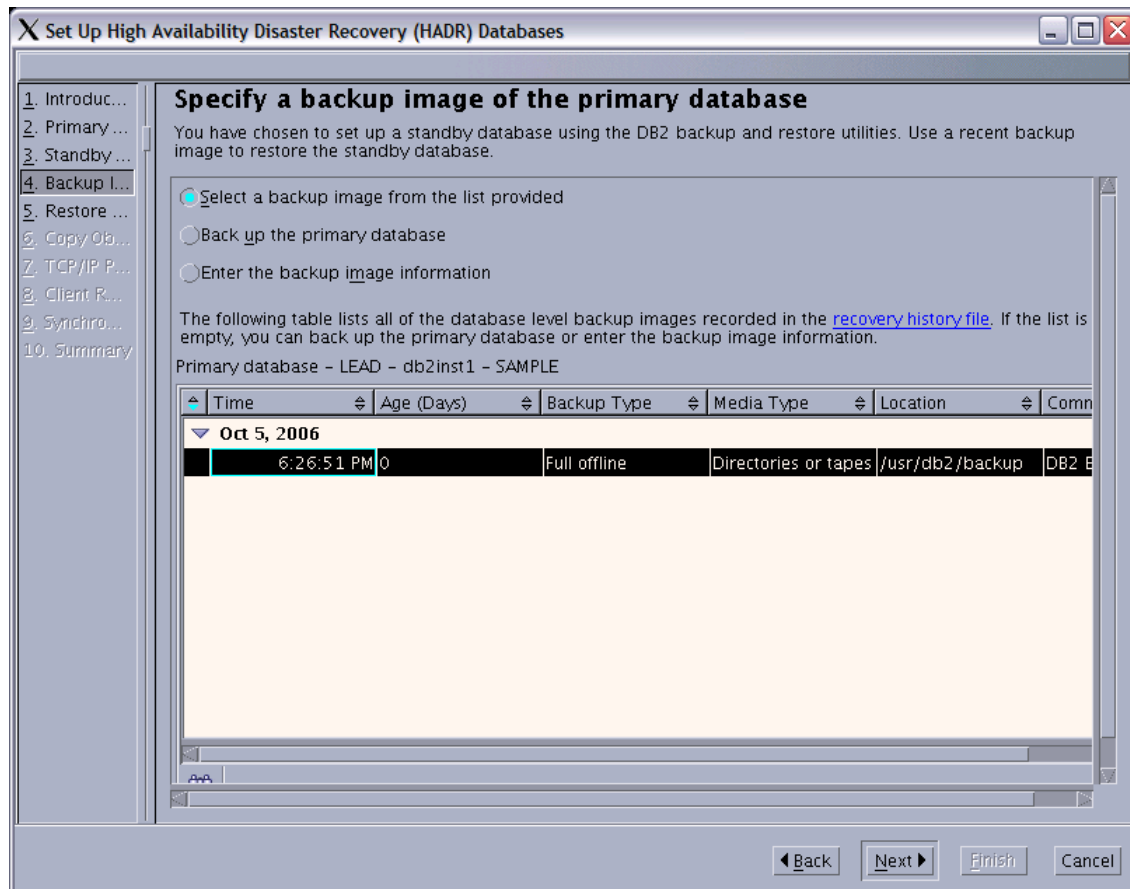


Figure 3-6 Selecting the preferred primary backup image to use for a restore on the standby server

7. *Restore database on standby system*: In Figure 3-7, specify an alias for the standby database on the remote system (POLONIUM) to be used by the primary instance on the local system (LEAD). This should exist from the time the standby database was identified in Figure 3-4.

For this example, we test the ability of DB2 file transfer by selecting **Copy the backup image from the primary system to the standby system**. Specify the location on the standby system where the backup file will be transferred by the wizard. Click **Next**.

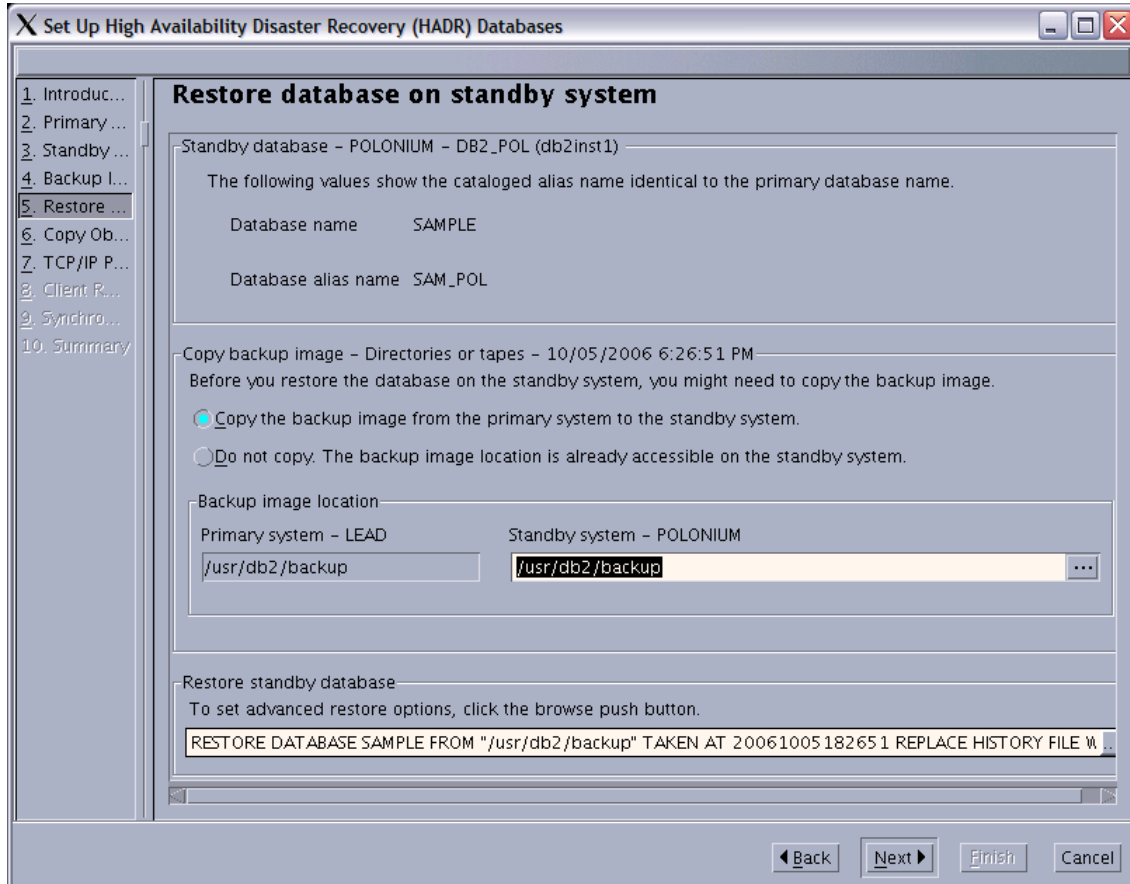


Figure 3-7 Restoring the primary backup image across to the standby server

8. *Copy objects to the standby system* (see Figure 3-8): If you have DB2 objects that do not get copied inside a database backup file, such as external code for UDFs and stored procedures, you can identify those objects here and have DB2 move them for you. Since we had none of those, we left the dialog box blank. Click **Next**.

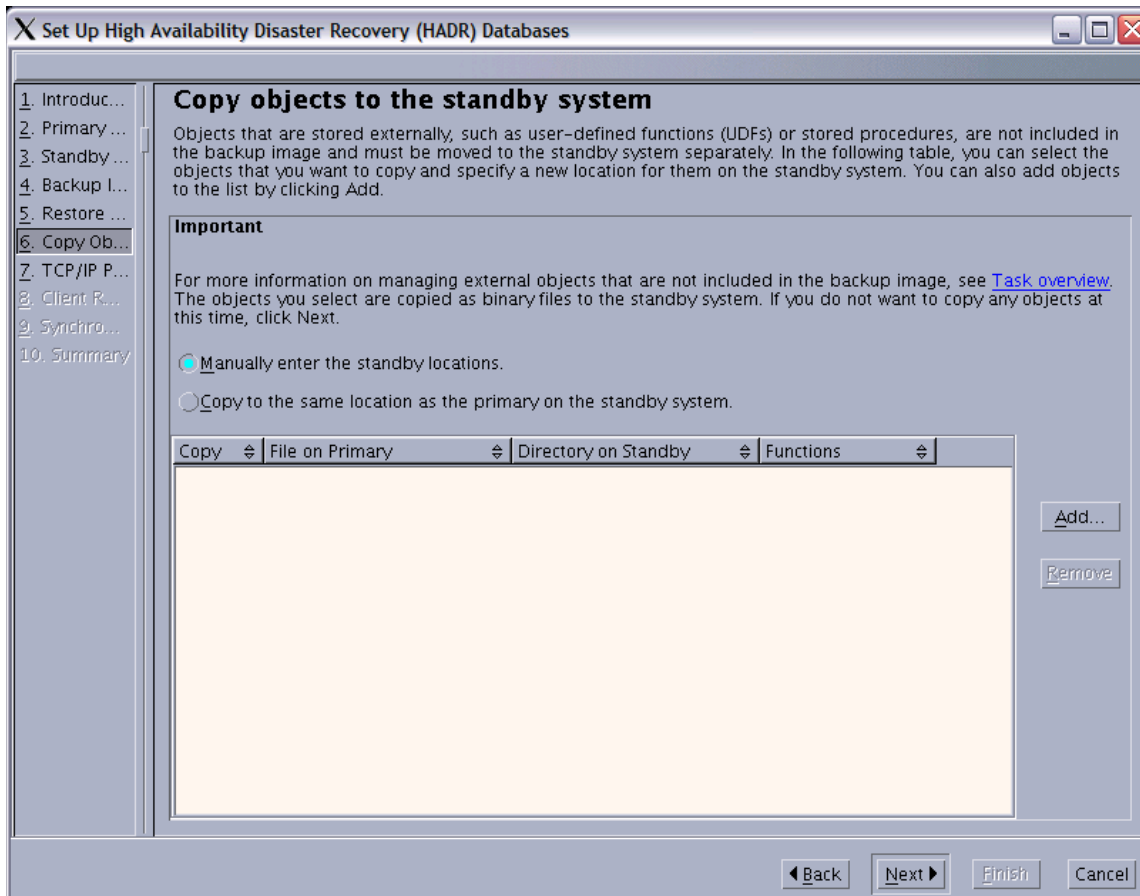


Figure 3-8 Copy “special” objects across to the standby system

9. *Specify TCP/IP communication parameters:* Figure 3-9 shows the settings required for the HADR-specific host name and service/port number, for the primary and standby databases. Enter unique values as required and click **Next**.

**Tip:** The wizard will automatically insert values for the HADR services in the `/etc/hosts` file. If you run the wizard more than once or have these services already registered, the service port numbers in this dialog box will be incremented (for example, 55003 and 55004, if we ran this same wizard again after using 55001 and 55002 the first time). Ensure that you manually change the values back as necessary on subsequent runs.



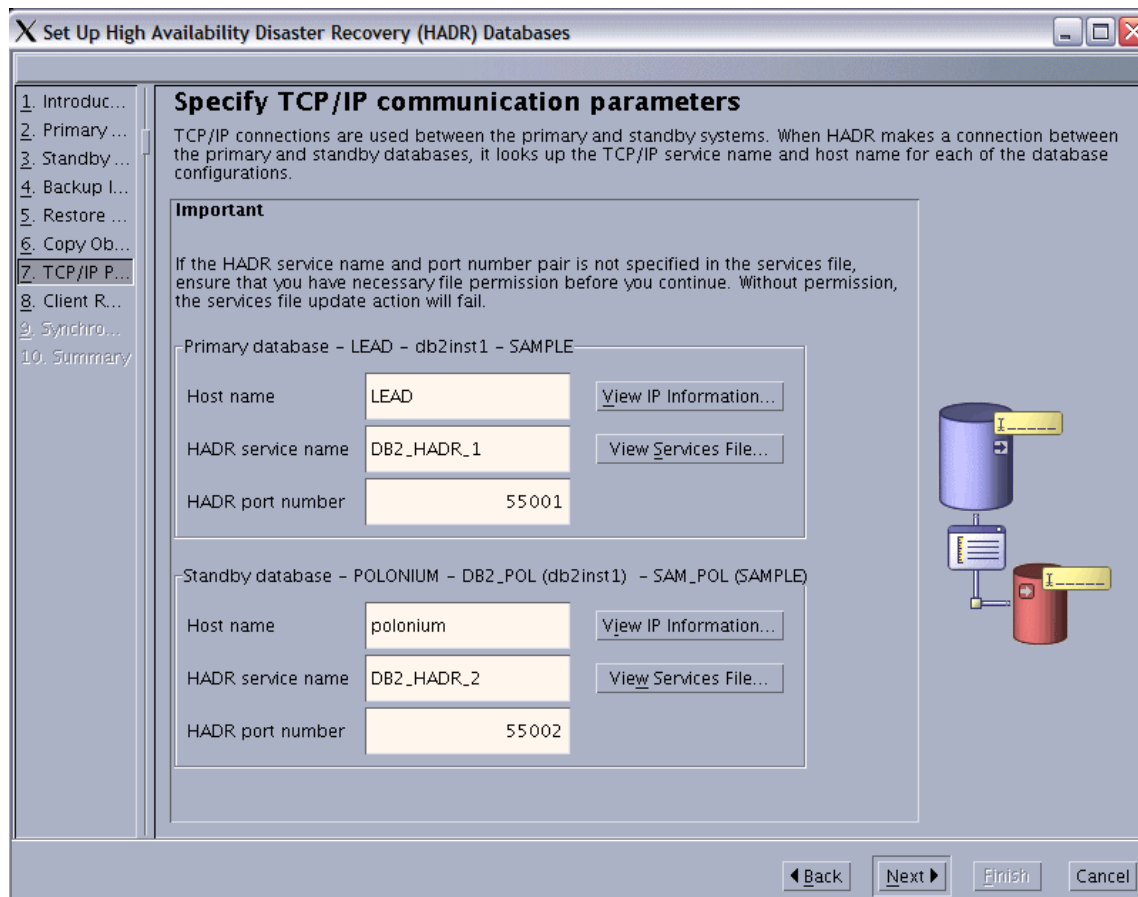


Figure 3-9 Specify host addr and port numbers for both HADR communication channels

10. *Configure databases for automatic client reroute*: Figure 3-10 shows where we set up the IP address redirect facility: where each client connection to a DB2 server stores the primary IP address and port number of the primary server, and also of the standby server. When you switch roles, or force a takeover, the other IP address is then used by the client; all the client sees is an SQL message to inform that the connection has changed and the transaction or database connection should be retried. See Chapter 5, “Automatic client reroute” on page 115 for more details on ACR.

The port number refers to the main port number for the DB2 instance, not to the HADR port. Just match the port number with the correct server name/IP address, and remember that the “alternate” for the primary is the standby server, and vice versa. Click **Next**.

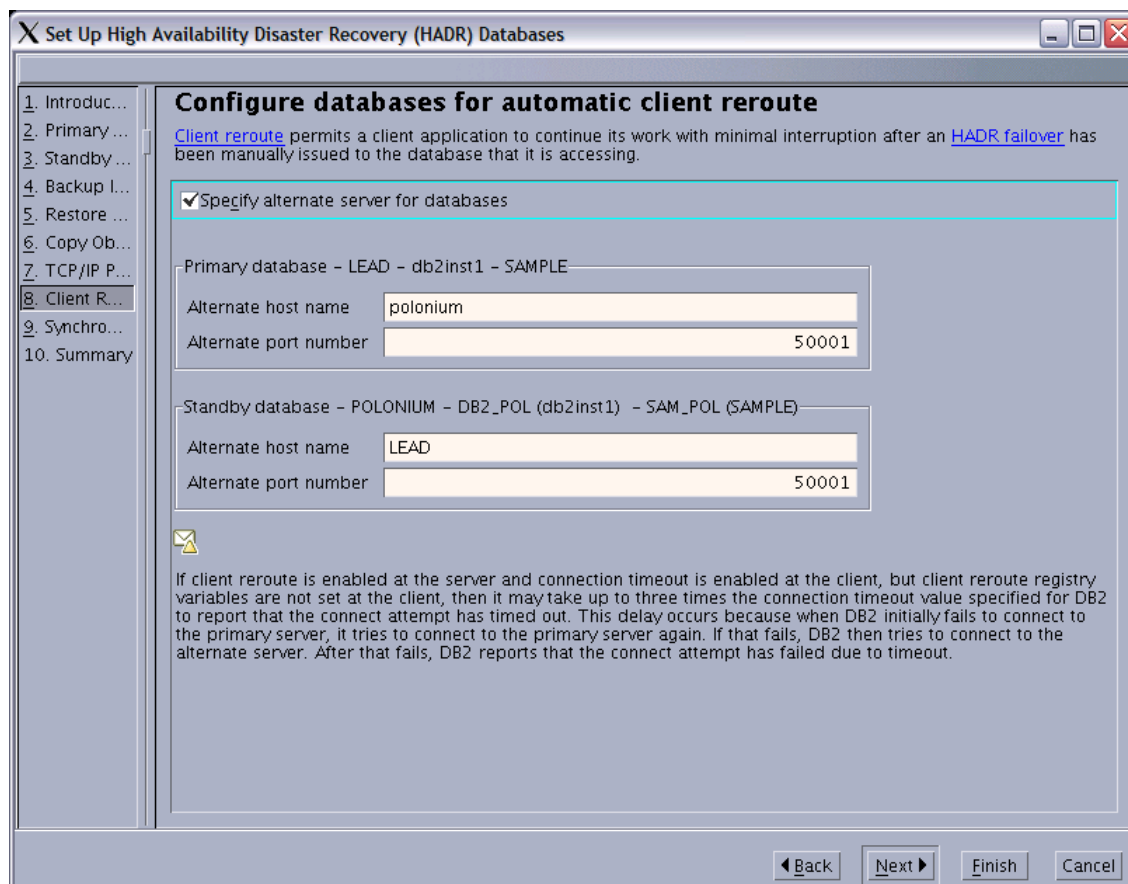


Figure 3-10 Configuring Automatic Client Reroute server and port information

11. *Specify synchronization mode for Peer state log writing:* Figure 3-11 lets us choose whether we want Synchronous, Near-Synchronous, or Asynchronous mode for HADR. We choose Synchronous for our example. See 2.4, “HADR architecture” on page 30 for more information on the three available synchronization modes. Specify the timeout value. For details of the best practice on choosing the synchronization mode and recommendations on how to speed up takeover time, see Chapter 6, “HADR best practice” on page 137.

Click **Next**.

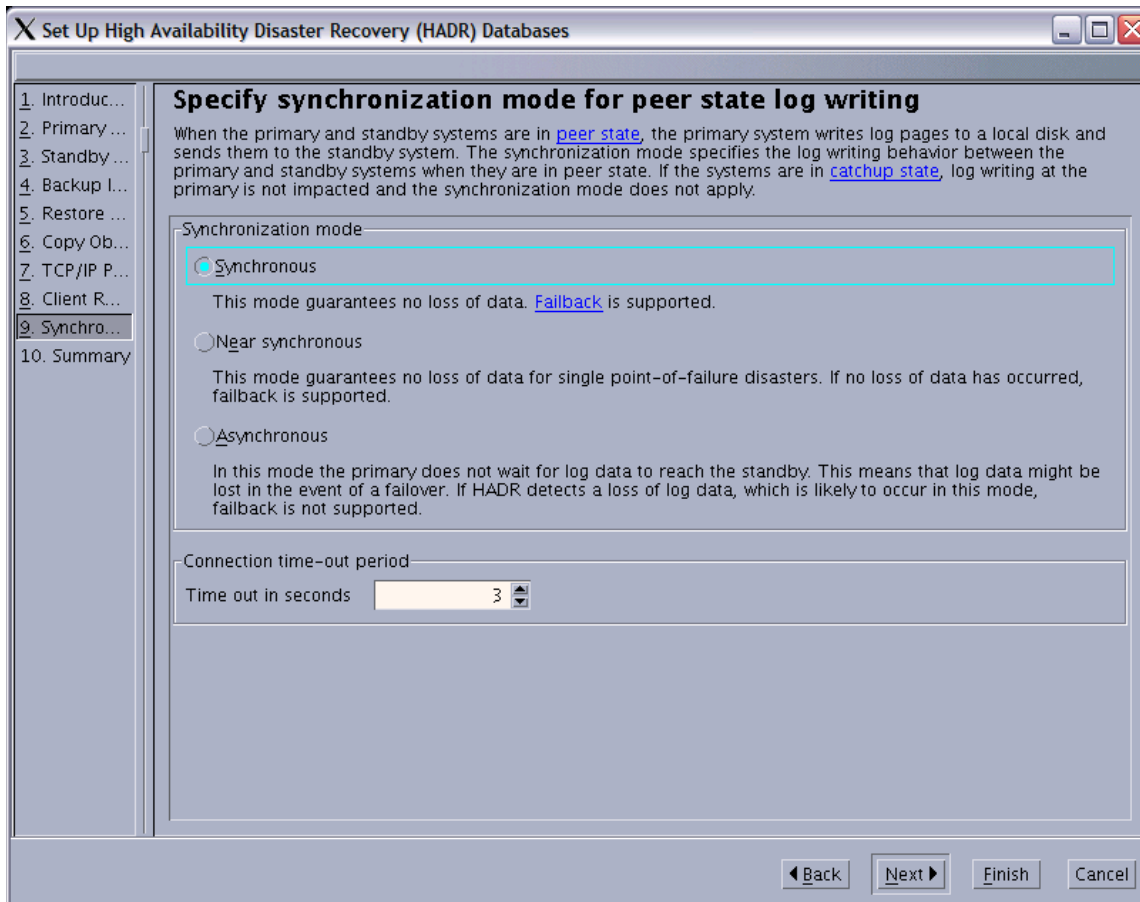


Figure 3-11 Choose an HADR syncmode: SYNC, NEARSYNC, or ASYNC

12. *Review the actions that will occur when you click Finish* (Figure 3-12):  
This dialog box shows an overview of the settings chosen for this HADR configuration. Click **Show Command...** to get command line statements in order to learn what goes on behind the scenes in the wizard. These commands are worth saving in a text file, certainly for subsequent configurations.

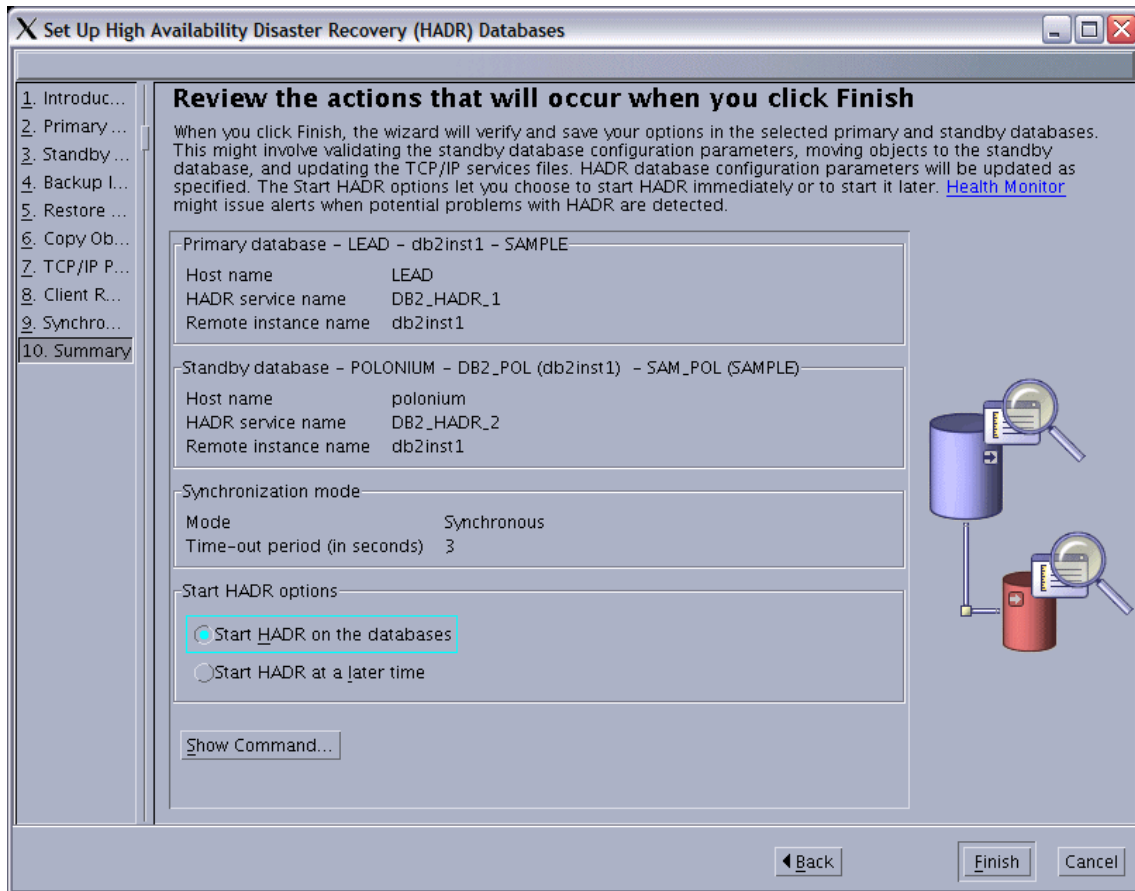


Figure 3-12 Reviewing the proposed configuration before commencing final HADR scripted steps

You can choose to either have the wizard start HADR, or issue that command yourself afterward through the Manage HADR wizard or on the command line.

Click **Finish**.

**Note:** The Show Command only takes into consideration the work to be done after configuring linear/archival logging, so prerequisite updates to the database configuration parameters LOGARCHMETH1/2 and LOGINDEXBUILD will not be listed.

13. *Execution of Steps*: Figure 3-13 shows a dynamic summary of those commands being generated and run by the automated script. If any issues are encountered, the wizard should let you know what must be corrected, and you should be able to click **Finish** on the previous dialog box in order to retry the process.

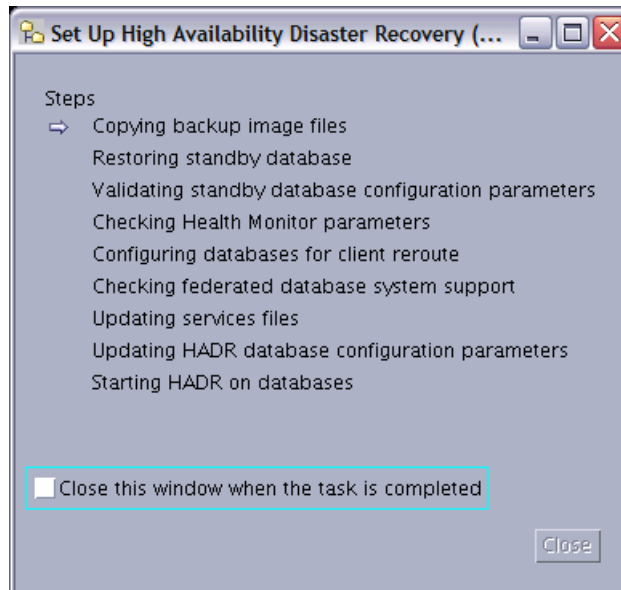


Figure 3-13 Commencing the steps to process a scripted HADR setup

Figure 3-14 shows the completed steps, which is what you should expect. HADR setup is now complete. Click **Close**.



Figure 3-14 The completed steps for a scripted HADR setup

14. *Verify HADR setup:* To confirm that HADR is up and running, you can check with the DB2 Control Center.

First, select **View** → **Refresh**, otherwise it will display the remote database as “unknown”. Next, right-click either of the HADR databases and select **High Availability Disaster Recovery** → **Manage HADR**. The result should be similar to Figure 3-15.

If you left the choice to start HADR as blank in step 11 (Figure 3-12 on page 62), you can click **Start HADR...** here, which should then show your two databases in a Peer state, with the appropriate synchronization mode, the connected state, and the log information on each. You can get similar information using the GET SNAPSHOT command or db2pd.

**Note:** When using DB2 Version 8, we recommend at least FixPak 14 to correct SYNCMODE value discrepancies in the Manage HADR dialog.



Figure 3-15 In Control Center: Manage HADR

### 3.2.3 Command line setup

DB2 provides both commands and GUI setup wizard for HARD. In this section, we demonstrate how to set up HADR using commands. In our Lab environment, we set up the LEAD server as the primary server, and POLONIUM server as the standby with the database SAMPLE on both the servers.

To set up HADR for the command line environment, complete the following steps:

1. Set the required database configuration parameters.

If archive logging is not already turned on, then update the LOGRETAIN parameter using the following command:

```
db2 update database configuration for sample using LOGRETAIN  
recovery
```

Also, set the LOGINDEXBUILD parameter, so that index creation, re-creation, or reorganization operations are logged, using the following command:

```
db2 update database configuration for sample using LOGINDEXBUILD ON
```

2. Back up your database on the primary and move a copy to the standby server using the following command:

```
db2 backup database sample to /usr/db2/backup
```

3. Restore the database on your standby system.

In our example, we restore to the SAMPLE database on the POLONIUM server using the following command:

```
db2 restore database sample from /usr/db2/backup taken at  
20061011141321 replace history file
```

4. Configure databases for ACR. This step is optional but highly recommended. To configure ACR, update the ALTERNATE SERVER database configuration parameter on both the primary and standby servers with the following steps:

- a. On the primary server, LEAD in our test case, set the standby server as the alternate server using the following command:

```
db2 update alternate server for database sample using hostname  
polonium port 50001
```

- b. On the standby server, POLONIUM in our test case, set the primary server as the alternate server using the following command:

```
db2 update alternate server for database sample using hostname  
lead port 50001
```

5. Update the following fields in the services file on the primary system LEAD for HADR communication:

- Service name: DB2\_HADR\_1
- Port number: 55001
- Service name: DB2\_HADR\_2
- Port number: 55002



6. Update the following fields in the services file on the standby system POLONIUM for HADR communication:
  - Service name: DB2\_HADR\_1
  - Port number: 55001
  - Service name: DB2\_HADR\_2
  - Port number: 55002
7. Update HADR database configuration parameters on the primary database with the following commands. In our test case, it is LEAD.
 

```
db2 update db cfg for sample using HADR_LOCAL_HOST LEAD
db2 update db cfg for sample using HADR_LOCAL_SVC DB2_HADR_1
db2 update db cfg for sample using HADR_REMOTE_HOST POLONIUM
db2 update db cfg for sample using HADR_REMOTE_SVC DB2_HADR_2
db2 update db cfg for sample using HADR_REMOTE_INST DB2INST1
db2 update db cfg for sample using HADR_SYNCMODE SYNC
db2 update db cfg for sample using HADR_TIMEOUT 3
db2 connect to sample
db2 quiesce database immediate force connections
db2 unquiesce database
db2 connect reset
```
8. Update HADR database configuration parameters on the standby database, POLONIUM in our test case, using the following commands:
 

```
db2 update db cfg for sample using HADR_LOCAL_HOST POLONIUM
db2 update db cfg for sample using HADR_LOCAL_SVC DB2_HADR_2
db2 update db cfg for sample using HADR_REMOTE_HOST LEAD
db2 update db cfg for sample using HADR_REMOTE_SVC DB2_HADR_1
db2 update db cfg for sample using HADR_REMOTE_INST db2inst1
db2 update db cfg for sample using HADR_SYNCMODE SYNC
db2 update db cfg for sample using HADR_TIMEOUT 3
```
9. Start HADR on the standby database on POLONIUM in our test case, using the following commands:
 

```
db2 deactivate database sample
db2 start hadr on database sample as standby
```
10. Start HADR on the primary database on LEAD in our test case, using the following commands:
 

```
db2 deactivate database sample
db2 start hadr on database sample as primary
```

## 3.3 Basic operation

This section give you basic details on the start, stop, and takeover operations preformed for HADR. We give examples for both the command line environment and GUI environment.

### 3.3.1 Starting up and shutting down

This section covers the startup and shutdown procedures for HADR on a GUI and command line environment.

#### Startup

Before starting HADR, you should ensure that your database manager (instance) on both the primary and standby databases is started. Use the **db2start** command to start the instance. The instance can be started in any order, primary or standby.

When starting HADR, we recommend that you start the standby database before the primary database. The reason for starting the standby first is that the primary HADR startup, without the BY FORCE option, requires the standby to be active within the HADR\_TIMEOUT period or startup will fail to prevent a split-brain scenario.

#### *Using the startup command*

This is the syntax for the startup command:

```
START HADR ON DATABASE database-alias [USER username [USING password]]  
AS {PRIMARY [BY FORCE] | STANDBY}
```

When starting the primary, the BY FORCE option specifies that the HADR primary database will not wait for the standby database to connect to it. After a start BY FORCE, the primary database will still accept valid connections from the standby database whenever the standby later becomes available.

For example, to start HADR on the primary, issue the following command:

```
db2 start hadr on database sample as primary
```

It is important to note which database you are on when you start HADR.

You can find the details for the START HADR command in *DB2 9.1 Command Reference*, SC10-4226.

## Starting from the Control Center

HADR can also be started from the HADR wizard in the Control Center:

1. From the Control Center, expand the object tree down to the database object, right-click the database name you plan to start HADR on. Select **High Availability Disaster Recovery** → **Manage** as shown in Figure 3-16.

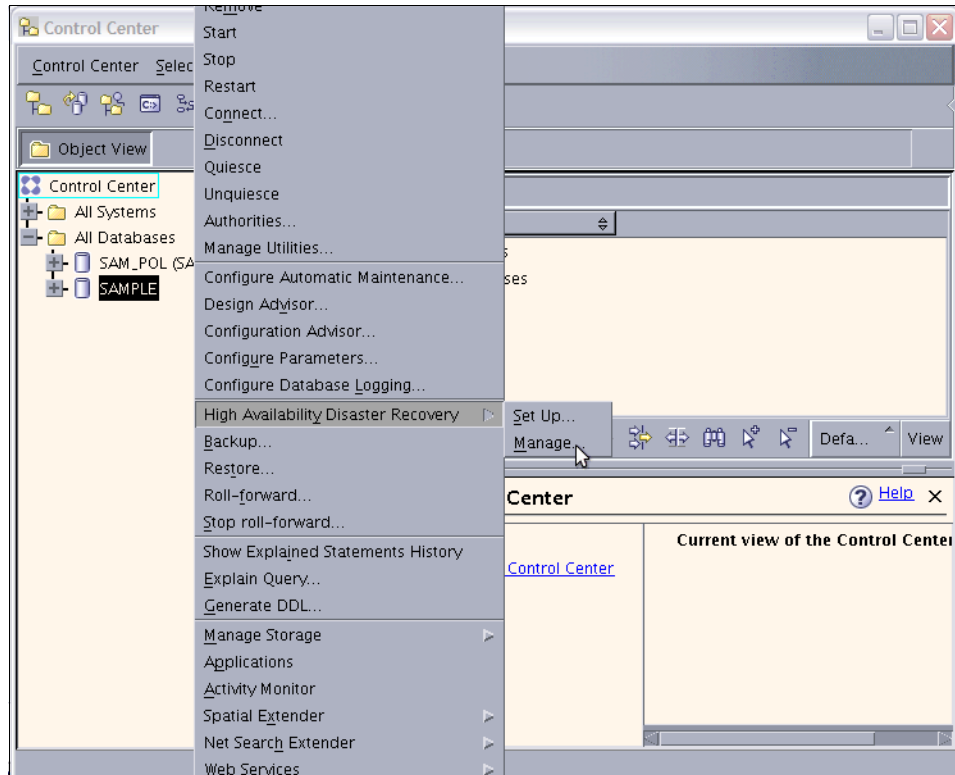


Figure 3-16 Manage HADR

2. From the Manage High Availability Disaster Recovery (HADR) window, click **Start HADR** as shown in Figure 3-17.

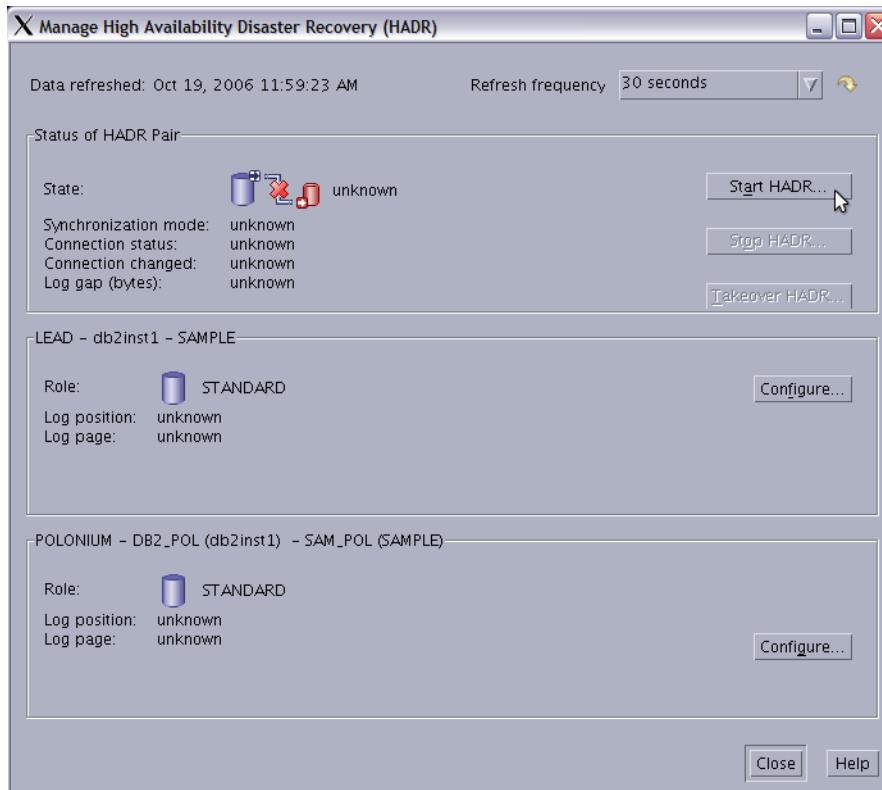


Figure 3-17 Start HADR from Manage screen

3. From the Start HADR window shown in Figure 3-18, you can choose to start the primary or the standby only, or both. Click **OK**.

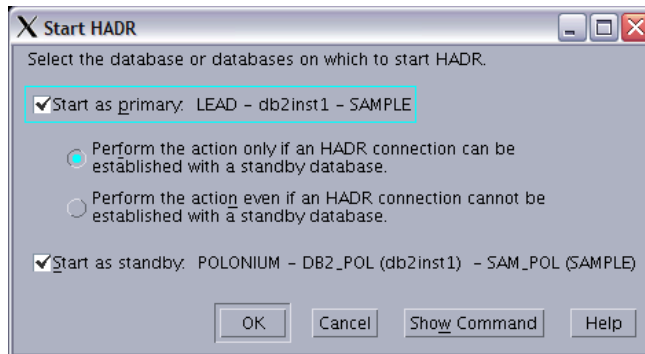


Figure 3-18 Start HADR

**Note:** When starting the primary using the choice, Perform the action even if an HADR connection cannot be established with a standby database, this issues the BY FORCE option in the START HADR command.

## Shutdown

Although the STOP HADR command can be used to stop HADR on the primary or the standby, or both, it should be used with caution. If you want to stop the specified database but still want it to maintain its role as either an HADR primary or a standby database, do not issue the STOP HADR command. If you issue the STOP HADR command, the database will become a standard database and might require re-initialization in order to resume operations as an HADR database. Instead, issue the DEACTIVATE DATABASE command.

If you only want to shut down the HADR operation, this is the recommended way of shutting down the HADR pair:

1. Deactivate the primary database.
2. Stop DB2 on the primary database.
3. Deactivate the standby database.
4. Stop DB2 on the standby database

### Using the STOP HADR command

You can stop HADR from the command line or the Control Center. When you want to bring your database to a standalone database, you can use the STOP HADR command.

The shutdown command has the following syntax:

```
STOP HADR ON DATABASE database-alias [USER username [USING password]]
```

For example:

```
db2 stop hadr on database sample
```

You can find the details for the STOP HADR command in the *DB2 9.1 Command Reference*, SC10-4226.

### ***Stopping HADR from the Control Center***

You can stop HADR with the HADR wizard by performing the following steps:

1. From the Control Center, expand the object tree down to the database object. Right-click the database name you plan to stop HADR on. Select **High Availability Disaster Recovery** → **Manage** (refer to Figure 3-16 on page 69).
2. From the Manage High Availability Disaster Recovery (HADR) window, click **Stop HADR** as shown in Figure 3-19.

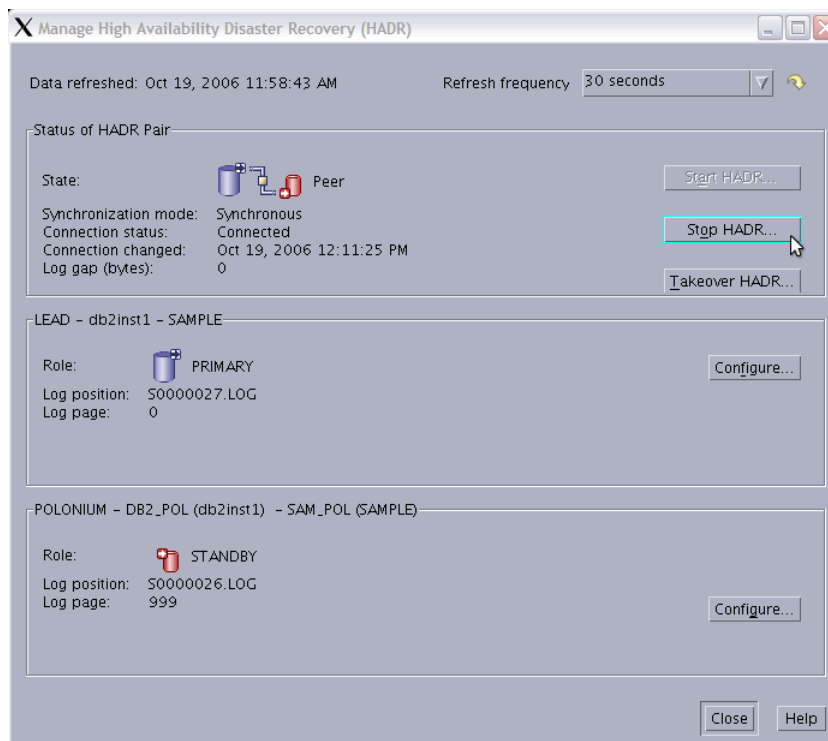


Figure 3-19 Manage HADR stop

3. From the Stop HADR window shown in Figure 3-20, you can choose to stop just the primary, or both the primary and the standby. If one of the databases has HADR stopped, the Stop HADR window only allows you to stop the database where HADR is started. Click **OK**.

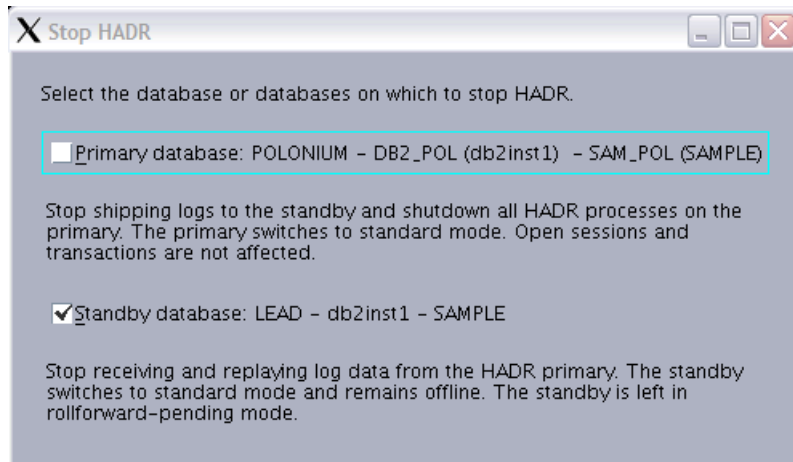


Figure 3-20 Stop HADR

### 3.3.2 Planned takeover

Planned takeover is also referred to as switching roles or role exchange. You issue the TAKEOVER command when you want to switch roles of the databases. For example, during an upgrade you switch roles, making the standby the new primary. Remember to re-route your clients either manually or by using the ACR after you have issued the TAKEOVER command.

Switching roles is only done from the standby when the databases are in the Peer state. The TAKEOVER command fails if the databases are in any other state.

The takeover procedure is simply issuing the TAKEOVER HADR command.

For example, issue the takeover command on the standby database using the following command:

```
db2 takeover hadr on database sample
```

The details for the TAKEOVER HADR command are found in the *DB2 9.1 Command Reference*, SC10-4226.

After issuing the TAKEOVER HADR command from the standby, the following steps are carried out in the background:

1. Standby tells the primary that it is taking over.
2. Primary forces off all client connections and refuses new connections.
3. Primary rolls back any open transactions and ships the remaining log, up to end of the log, to standby.
4. Standby replays received log, up to end of the log.
5. Primary becomes the new standby.
6. Standby becomes the new primary.

## Using the GUI

In our example, we start with SAMPLE database on POLONIUM as the primary database. Perform the following steps:

1. From the Control Center expand the object tree down to the database object, right-click the database name you plan to takeover HADR on. Select **High Availability Disaster Recovery → Manage**.



2. From the HADR window on the standby (POLONIUM), click **Takeover HADR** as shown in Figure 3-21.

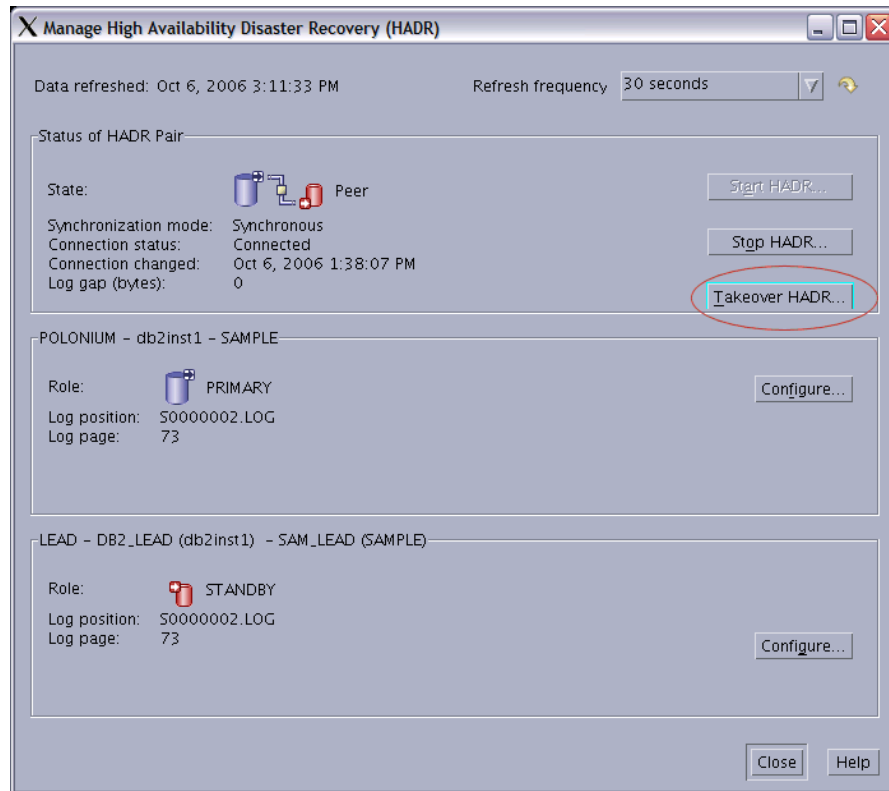


Figure 3-21 Manage HADR Takeover

3. From the Takeover HADR window, select **Switch roles** as shown in Figure 3-22 and click **OK**.

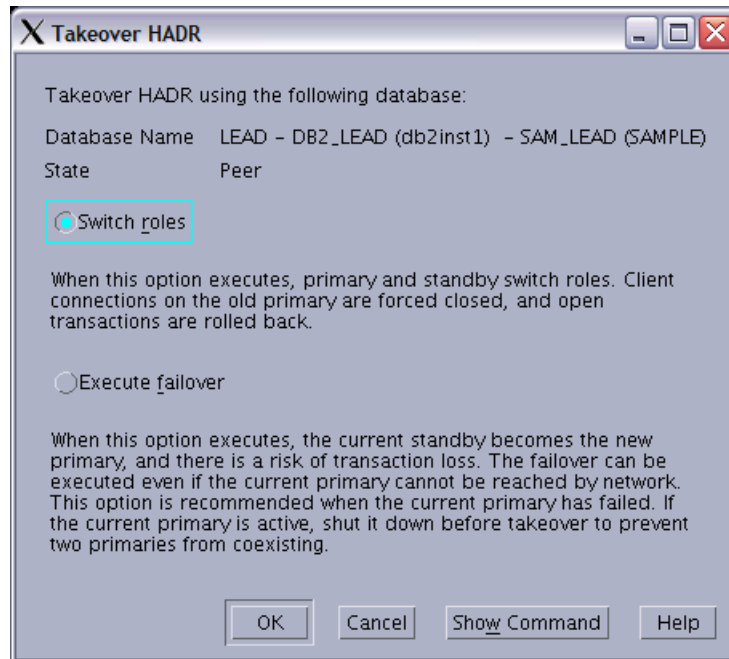


Figure 3-22 Takeover HADR

Clicking **Show Command** provides you the command DB2 used for this operation. Figure 3-23 illustrates the output.

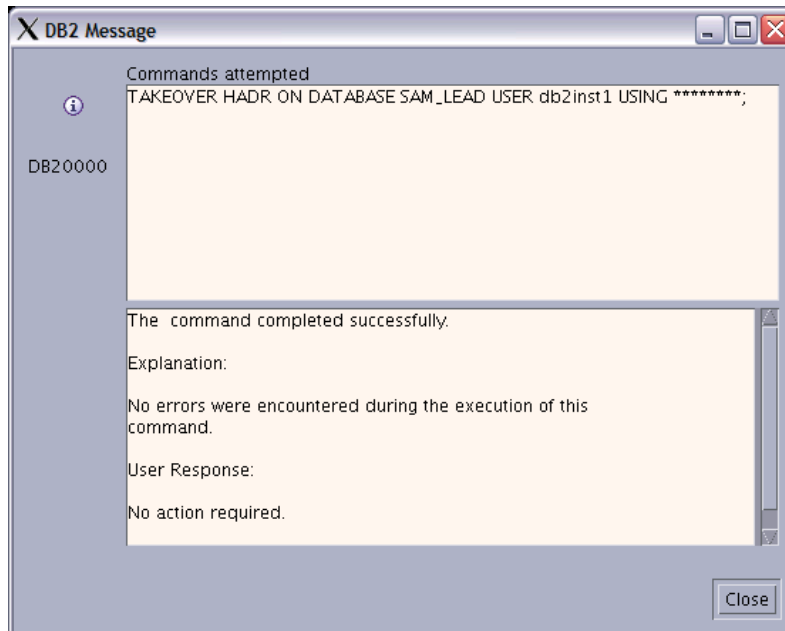


Figure 3-23 Takeover HADR command

Now the primary and the standby roles are swapped. Issue the same command again to swap back the roles. This can be done as often as necessary, so long as the databases are in the Peer state.

### 3.3.3 Takeover by force

A *takeover by force* is also referred to as *failover*, which is issued from the standby database with the TAKEOVER command BY FORCE option included. You should use the takeover by force only if your primary database is not functional. The BY FORCE option tells the standby to become the new primary without coordinating with the original primary as it does with a planned takeover.

The procedure for failover includes the following steps:

1. Make sure that the primary is down to minimize the chances of data loss. If a takeover BY FORCE is issued and the primary is not down, this can result in both the databases becoming primary databases. This is referred to as *split brain*.

2. Issue the TAKEOVER HADR command with the BY FORCE option on the standby database. The following is an example for the sample database:

**db2 takeover hadr on database *sample* by force**

After issuing the TAKEOVER HADR command with the BY FORCE option from the standby, the following steps are carried out in the background:

1. The standby sends a notice asking the primary to shut down.
2. The standby does *not* wait for any acknowledgement from the primary to confirm that it has received the takeover notification or that it has shut down.
3. The standby stops receiving logs from the primary, finishes replaying the logs it has already received, and then becomes a primary.

Data loss is possible when a takeover by force is issued. Your chance of data loss depends on your configuration and circumstances. The following list shows the general settings and the result of a failure on the primary.

- ▶ If the primary database is in the Peer state when it fails:
  - With syncmode set to SYNC, the standby will not lose any transactions that were reported committed to the application before the primary failed.
  - With syncmode set to NEARSYNC, the standby only loses transactions committed by the primary if the primary and standby databases fail at the same time. This is a highly unlikely circumstance.
  - With syncmode set to ASYNC, the standby database can lose transactions committed on the primary if the standby did not receive all of the log records for those particular transactions before the takeover operation was performed. As in the NEARSYNC mode, if the primary and the standby fail at the same time, transactions can be lost.
- ▶ If the primary is in the remote catchup pending state or any other non-Peer state when it fails:
  - For all the three syncmodes (SYNC, NEARSYNC, ASYNC), transactions that have not been received and processed by the standby database will be lost.

For more details on the ways to prevent data loss in a forced takeover, see Chapter 7 in *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228.

Following a takeover by force because of a failure at the primary database, once the old primary is recovered and you bring it back online, you can reintegrate the old primary as the standby database.

To reintegrate the old primary, perform the following steps:

1. Recover the failed primary, and bring it back online.
2. Restart the failed primary as the new standby using the START HADR command, for example:

```
db2 start hadr on database sample as standby
```

If the two databases have incompatible log streams, for example, because of logs not being received from the standby before takeover, then the reintegration of the old primary to the new standby will fail and you will have to restore a backup of the current primary to your failed primary to start it as the new standby. Refer to 3.4.5, “Re-establishing HADR after failure” on page 86 for more details on reintegration.

### **Example of takeover by force in GUI**

To perform a takeover by force in the GUI, complete the following steps:

1. From the Control Center, expand the object tree down to the database object, right-click the database name you plan to takeover HADR on. Select **High Availability Disaster Recovery** → **Manage** (refer to Figure 3-16 on page 69).
2. From the Manage High Availability Disaster Recovery (HADR) window on the standby (POLONIUM in our test case), click **Takeover HADR** as shown in Figure 3-21 on page 75.

3. From the Takeover HADR window, select **Execute failover** as shown in Figure 3-24 and click **OK**.

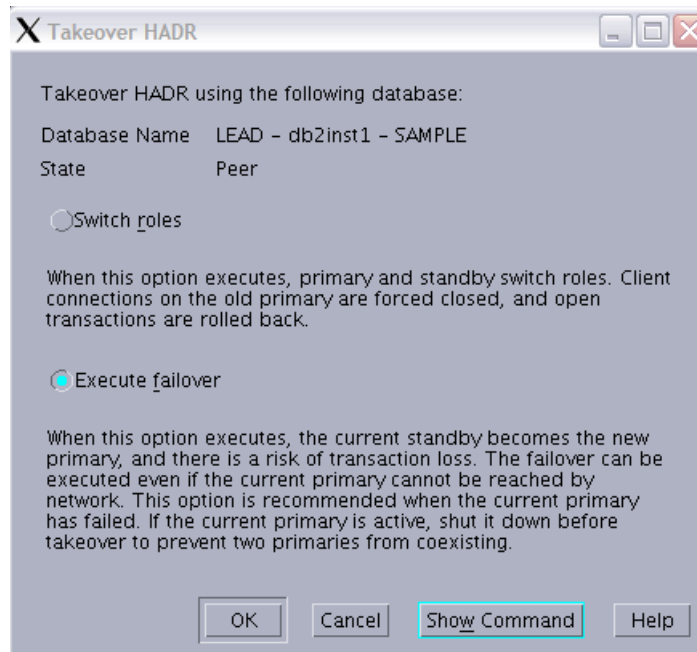


Figure 3-24 Takeover HADR by force

## 3.4 Troubleshooting

Here we examine some of the possible issues you might encounter while setting up and running HADR, and while attempting to resume the HADR operation after a server failure, or other unanticipated interruption to DB2 HADR communication. In most cases, the solutions and workarounds are straightforward, but not often immediately obvious from any error messages you might or might not be getting at the time. These examples are from first-hand experience, and the workarounds presented might not be the only way to resolve the situation.

An additional source of material to review in case you run into trouble when setting up, are the platform-specific prerequisites and maintenance dependencies which are dynamically updated online at the following Web site:

<http://www.ibm.com/software/data/db2/9/sysreqs.html>

[http://www.ibm.com/software/data/db2/9/sysreqs\\_v8.html](http://www.ibm.com/software/data/db2/9/sysreqs_v8.html)

### 3.4.1 During setup

Here is a list of common issues and possible fixes encountered during setup:

► Symptom:

On AIX/UNIX/Linux, error messages are indicating that graphical tools such as the Control Center cannot be started due to missing Java libraries.

– Fix:

For 64-bit DB2 V8, AIX system support might have to install 64-bit Java libraries, and then you set the admin cfg and dbm cfg value of JDK\_PATH, and admin cfg value of JDK\_64\_PATH appropriately, and restart the DB2 instance(s) and DB2 Admin Server.

DB2 9 comes with built-in Java libraries, so this problem should not occur.

► Symptom:

On Windows, various error messages are claiming that the user ID does not have the authority to execute the given command, even though the user ID is both the DB2 instance ID with implicit SYSADM authority, and a Windows Administrator.

– Fix:

As a prerequisite, ensure that if you are in a permanently connected domain, have the DB2ADMNS group with the DB2 instance user ID created at least on the domain and optionally replicated onto the local servers as a security policy. Ensure that you are logged on to Windows as the same user ID type as DB2 was installed with. In an Active Directory® configured server, this will most likely be the domain user ID rather than the local user ID. Ensure that the user ID you log on with is part of the Administrator's group. You can also encounter issues with the DB2 instance user ID's implicit authority if you change the SYSADM\_GROUP db cfg parameter from the default blank value, even if the DB2 instance ID is connected to the SYSADM\_GROUP.

If you are running DB2 on a server that is configured in a domain, but is not permanently connected to that domain and does not have a security policy periodically overwriting the local registry, then it might be better to install everything using the local user ID. A final consideration is to check the value of the DB2 registry variable DB2\_GRP\_LOOKUP, which is specific to the Windows environment. Users who log on as the domain user ID have their group connections in Windows checked against the domain groups by default, rather than local groups. Setting the DB2\_GRP\_LOOKUP parameter to LOCAL overrides this behavior.

► Symptom:

You cannot see the remote server or DB2 objects on the remote/standby end when attempting to discover DB2 instances and databases.

– Fix:

Start DB2 Admin Server on the remote end, and confirm firewall is open for port 523, and/or manually catalog the administrative TCP/IP node, the instance TCP/IP node, and the remote database on the command line. See Example 3-4. If you wish to manage or monitor HADR via the Control Center, you will need the Admin Server to be working at both ends for connectivity.

*Example 3-4 Catalog TCP/IP node and database*

---

```
db2 catalog admin tcpip node polonium remote x.x.x.x remote_instance
db2inst1 system polonium
db2 catalog tcpip node db2_pol remote x.x.x.x server 50001
remote_instance db2inst1 system polonium
db2 catalog database sample as sam_pol at node db2_pol
```

---

If you execute the commands to catalog nodes and database manually in the middle of the Setup HADR dialog boxes, you need to cancel out to the Control Center main dialog box, select **View** → **Refresh**, and restart, in order to see the remote objects in the dialog boxes.

### 3.4.2 After setup or during normal execution

The following common issues and possible fixes are encountered after setup and when HADR is operating:

► Symptom:

Cannot see the remote server or DB2 objects on the remote/standby end in the Control Center Manage HADR dialog box. The values are displayed as “unknown”, even when DB2 on the remote server is definitely up and running, and HADR status displays as connected and in the Peer state in output from **db2pd** or **get snapshot for db** commands.

– Fix:

- i. The first thing to check is that DB2 Admin Server is started on the remote server.
- ii. Next, confirm that the ADMIN TCPIP and TCPIP NODEs and remote database are cataloged correctly.
- iii. Ensure that the server name or IP address as specified in the cataloged node names is actually registered in both the local and the



remote /etc/hosts files, and that they match the host name or IP address as registered in the remote server's <DB2 Instance home>/sqllib/db2nodes.cfg. By experience we have found that servers on VM or similar architectures can have multiple IP addresses defined, only some of which are visible through firewalls, and which might not match the host name a remote administrative client should be able to see.

► Symptom:

SQL30081N Communication error has been detected when attempting to connect to the database on the remote server, even though previously there was no issue, and both the relevant DB2 instance and DB2 Admin Server are definitely up and running on the remote server.

— Fix:

This is an extremely common issue on a dynamic network environment, and could be caused by any number of reasons, including firewall issues, IP address or port number mismatches or conflicts, incorrect /etc/hosts or /etc/hosts.allow/deny content, host name or DNS issues. The first thing to do is to try whatever the Messages Reference recommends for any protocol specific error codes (refer to the *DB2 9.1 Message Reference Volume 2*, SC10-4239, Chapter 2). After exhausting that, you might try falling back from usage of DNS addresses and service names, and temporarily use the direct IP address and port numbers instead, which you know to be correct and open in the firewall. If this issue persists, ask whomsoever supports the network architecture to check that the firewall ports are open and whatever else might be causing the described communication error symptoms, providing them with all the protocol specific error message information.

► Symptom:

Synchronization mode on the Manage HADR dialog box does not match the output from **db2pd**, **get db cfg**, or **get snapshot for db**. NEARSYNC appears correctly, but SYNC and ASYNC appear to be swapped.

— Fix:

This is a known cosmetic error isolated to the Manage HADR dialog, which has now been corrected. We recommend using DB2 V8 FixPak 14 in order to display the correct SYNCMODE value.

► Symptom:

**db2pd** output does not work on DB2 V8 FixPak13.

— Fix:

This is a known problem only at this specific fix pack level, and at the time of writing, open APAR JR24668 should eventually address this. The

work-around is to use **db2 get snapshot** or other means to access the required information.

► Symptom:

After changing db cfg parameters on the both servers, performing deactivation and reactivation on the standby, HADR will still not reconnect, so that roles can be switched for the second database deactivation/reactivation:

— Fix:

If you are changing certain parameters, such as any of the HADR\_ prefixed db cfg parameters, the currently active values must match on both sides in order for HADR connectivity to work. You will need downtime on the primary after changing the db cfg parameter on both the servers to establish the new value as current. Plan ahead for an outage window to achieve this, otherwise you risk rendering your standby inoperative and disconnected until you change the db cfg value back again to match the current value on the primary.

### 3.4.3 After an HADR disconnect or server failure

Here is an issue and possible fixes encountered after HADR is disconnected or a server failure.

► Symptom:

As an example, let us say an operating system fix is carried out, and the server gets recycled without so much as a courtesy e-mail to warn DB2 support about it. As a result, HADR becomes disconnected, and attempts to restart it continually fail.

— Fix:

In this case, the cause will most likely be mismatching or lost log records on the standby, such that it cannot reintegrate successfully with the primary. You can find this out by just looking at the output in the db2diag.log file for the times you attempt to start HADR on either the primary or the standby. The error messages for this are clear. Subsequent corrective action is straightforward.

You have no choice but to re-establish HADR by backing up the primary, transferring it to the standby, and restoring it there. Instructions for this are fully set out in 3.4.5, “Re-establishing HADR after failure” on page 86.

### 3.4.4 Considerations while running HADR

Once the HADR is configured and running, consider these situations:

- ▶ If you schedule backup and housekeeping tasks through the DB2 tools database, you should consider managing it through HADR in parallel with your application database. In the case where an extended outage of the primary server carries over through a critical batch window, you will want it to be working and running the same tasks just as on the original primary server.
- ▶ The db cfg parameter AUTORESTART should be turned off if you do not want the standby database to attempt to roll forward through logs while HADR is stopped and the DB2 instance gets recycled while that database is still in STANDARD mode.
- ▶ If you use ACR rather than a clustering solution to control IP addressing and takeover for remote clients, **db2iauto** should be turned off for the DB2 instances which control HADR databases. To explain using an example of server A and server B, this is so that DB2 on a broken primary server A does not automatically restart when server A is subsequently fixed and started up, causing a split-brain scenario.

In the context of HADR, *split brain* can be summarized as where remote clients cannot distinguish one server from the other when connecting to their database, opening the possibility of changes being made to the wrong database, and then being unable to be isolated and transferred to the correct database.

Fortunately, DB2 HADR is able to protect itself from split-brain scenarios where an HADR primary database has been deactivated while it was in HADR primary mode. If a client attempts to connect to such a database, it will go through an attempt to activate the database in HADR primary mode, and connect to an HADR standby. After the HADR\_TIMEOUT interval, DB2 will return an SQL1768N with reason code 7:

The primary database failed to establish a connection to its standby database within the HADR timeout interval.

This is also why it is easier from an Administration point of view to start the HADR standby database first, then the primary, as there is no pressure to get the other database started before the HADR\_TIMEOUT interval lapses and the database is left inactive again.

Split brain in the context of HADR can still occur in other ways:

- Where the broken HADR primary server was never actually deactivated (for example, where the failure was with the network rather than with the server), and after a **db2 takeover hadr on database ... by force** command is issued by the standby server B, the broken primary server A unexpectedly comes back online.

A proper clustering solution should be able to cater to this problem by use of resource group takeover actions, where a primary server which finds itself isolated will know to relinquish ownership of resources and issue shutdown scripts to critical resources on that server, but there is no easy fix in a non-clustering scenario. This can also happen irrespective of any temporary failure — for example, when the forced takeover command is issued on server B but the HADR primary on server A was never stopped.

- Where the broken HADR PRIMARY on server A has a **db2 stop hadr on database** command issued on it, such that it is now in STANDARD mode, and clients can once again connect to it instead of the correct HADR primary database running on server B. As with the previous scenario, this should not be possible in a proper clustering solution that manages IP addressing. Remote clients in this case can only connect to a single server that has the role of the active node.

### 3.4.5 Re-establishing HADR after failure

Also referred to as *reintegration* of HADR, this process is very similar to the initial setup of HADR, but without all the configuration of db cfg parameters.

There are two ways to successfully get HADR working again in a paired primary and standby relationship; both should require no downtime, and the first is an extremely rapid and low complexity solution.

For familiarity, we are using the example of HADR database SAMPLE on both Server A as primary and Server B as standby, where Server A suffers a failure, followed by a **takeover hadr by force** command on server B, such that server B becomes the new primary.

The first way will only work if the primary and the standby were in Peer state at the time of server A's failure. In this scenario, the DB2 instance is eventually restarted on server A after correcting whatever failure occurred. DB2 HADR will protect itself from split brain by not allowing activation of the database on server A, returning SQL1768N with RC 7 if any database activation or remote client connection attempts are made at this time.

The next thing to do is to issue the command **db2 start hadr on database sample as standby** on Server A. Note that database SAMPLE also cannot have had a **db2 stop hadr on database sample** command run beforehand, as it will then contain log records that server B does not know about, making it effectively unusable as a standby for server B. Attempts to restart HADR on it as standby will result in SQL1767N RC 1. Attempting to start HADR as standby on server A's SAMPLE database will naturally fail for the same reason if the databases were not in Peer state at the time of server A's failure.

In an ideal situation, the Peer state should be continually tracked on the HADR primary server. If any failure occurs while not in Peer state, then consideration should be made before issuing any **db2 hadr takeover database sample by force** command on the standby server B. In this situation, every effort should be made to correct issues on Server A and have that database restarted rather than issue a takeover by force command on Server B's STANDBY database, as Peer state is the only guarantee you have that there has been no lost data from committed transactions.

The second way will work regardless of whether the HADR pair was in Peer state at the time of a server failure. This method is relatively less complex, and saves time compared to the initial setup of HADR, since the db cfg parameters are stored separately from the physical data containers, in a non-user-readable file called SQLDBCONF. This file is not replaced by a restore command unless that database has been dropped prior to the restore.

At the most, if you need to drop your standby database before executing the restore from primary backup, you would need to change the following db cfg parameters:

- ▶ HADR\_LOCAL\_HOST = <hostname>
- ▶ HADR\_LOCAL\_SVC = <this server's HADR port# >
- ▶ HADR\_REMOTE\_HOST = <the remote server hostname>
- ▶ HADR\_REMOTE\_SVC = <the remote server's HADR port#>
- ▶ HADR\_REMOTE\_INST = <the remote server's DB2 Instance name>

Normally, you would never need to drop your standby database prior to the restore. However, DB2 will not let you restore over the top of a database which is currently in an HADR standby or primary state, and in some cases, DB2 will not let you stop HADR and put that database into a standard mode; this can happen if HADR was not stopped on a database before the DB2 instance is migrated from version 8 to version 9 for example.

Apart from not having to change your db cfg parameters after a restore, one reason you would not want to drop the database before the restore would be if you have a database with large containers, irrespective of the amount of data contained within them. DB2 needs time to pre-format at the time of database creation. Large containers will take more time. The time needed depends on the storage and system capacity. Restoring the database without dropping the database beforehand would save the pre-format time.

If you are concerned about scheduling a downtime window for re-establishing HADR, you can rest assured that you should not need one: at most, you might experience some read-only impact for a short duration while the online backup takes place on the primary database. As *DB2 9.1 Command Reference*, SC10-4226, Chapter 3 states:

“During an online backup, DB2 obtains IN (Intent None) locks on all tables existing in SMS table spaces as they are processed and S (Share) locks on LOB data in SMS table spaces. “

## Command line steps

Re-establishing (reintegrating) the standby with the primary can be achieved through a few command line steps issued by the DB2 instance user ID:

1. Prepare a database backup to restore the broken standby with:

Backup your primary database, online or offline, it does not matter. You might wish to compress the backup to save time when transferring it to the remote (standby) server, or use a TSM target which the remote server has been authorized to read from. In our example on Linux, we make a compressed backup of the SAMPLE database on the local server (LEAD) to local disk, then use **scp** to transfer the backup file to the remote server (POLONIUM):

```
db2 backup database sample online to /usr/db2/backup compress
```

2. Transfer your primary database backup file to the remote server:

For our example, we need to find the name of the backup file; the destination of the backup step was /usr/db2/backup. In Windows, up to and including DB2 V8, the file name of the backup files consists of a hierarchy of subdirectories inside the destination subdirectory. The file name to be copied is only the final portion of the hhmmss portion plus a three digit extension indicating the backup sequence number, usually .001, unless you perform concurrent backups. As of DB2 9, the backup file name on Windows matches Linux/UNIX, a long file name in a single subdirectory.

In Example 3-5, Linux server, **ls -l** of **/usr/db2/backup** shows:

### Example 3-5 List database backup files

---

```
db2inst1@lead:/usr/db2/backup> ls -l
total 53380
-rw-r----- 1 db2inst1 db2grp1 12603392 2006-10-05 18:26 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061005182651.001
-rw-r----- 1 db2inst1 db2grp1 12079104 2006-10-11 13:46 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061011134630.001
-rw-r----- 1 db2inst1 db2grp1 12079104 2006-10-11 14:13 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061011141321.001
-rw-r----- 1 db2inst1 db2grp1 17846272 2006-10-18 15:11 SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001
```

---

The backup file name we want will contain a matching timestamp with the output of the above example backup command as follows:

Backup successful. The timestamp for this backup image is : 20061018151120

The matching file for us is:

SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001

The timestamp (20061018151120, in our example), is also used in the restore command later, so note it down.

Now we actually issue the transfer command, whether that be **scp** as shown in our example, or **sftp** or some other method:

```
scp SAMPLE.0.db2inst1.NODE0000.CATN0000.20061018151120.001
db2inst1@polonium:/usr/db2/backup
```

3. We log on to the remote server (POLONIUM in our example) and issue the restore database command. Note that HADR will not be able to re-establish a connection if the database on the standby end is rolled forward after the restore:

```
db2 restore db sample from /usr/db2/backup taken at 20061018151120
```

As you can see, we use the timestamp in the restore command. Respond “y” to any prompt to overwrite an existing database.

4. The final step simply involves starting HADR on both ends, and checking it is connecting successfully. You can wait around until the standby database reaches the Peer state, or just leave it in catchup-pending, where it will eventually catch up and reach the Peer state.
  - a. In our example, the primary database never left HADR primary state, so we only need to start HADR on the standby:

```
db2 start hadr on db sample as standby
```

If HADR was not started on the primary server database, we could have achieved this by logging on there and issuing the following command:

```
db2 start hadr on db sample as primary
```

**Tip:** If HADR is stopped on both servers for a given database, restarting it should commence with the standby first, then the primary, otherwise you end up with this message:

```
SQL1768N  Unable to start HADR. Reason code = "7".
```

- b. Checking that HADR is connected and working should now be a matter of checking the **db2diag.log**, or **get snapshot for db**, or **db2pd** commands.

db2pd -db sample -hadr

Output for us is as follows in Example 3-6.

*Example 3-6 Checking HADR using db2pd output*

---

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:00:04

HADR Information:
Role      State              SyncMode HeartBeatsMissed   LogGapRunAvg (bytes)
Standby Peer              Sync        0                1381445

ConnectStatus ConnectTime              Timeout
Connected    Thu Oct 19 10:03:44 2006 (1161277424) 3

LocalHost                LocalService
polonium                 DB2_HADR_2

RemoteHost                RemoteService      RemoteInstance
LEAD                     DB2_HADR_1         db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000025.LOG 180      0x00000000071FC6F3

StandByFile StandByPg StandByLSN
S0000025.LOG 180      0x00000000071FC6F
```

---

- c. Optionally, after you reach the Peer state, you can switch roles. For instance, if you just restored the database on a broken server which was originally the primary system, and which is now running as standby, simply log on to that standby server and issue the non-forced takeover command:  
db2 takeover hadr on db sample



On our example environment, a subsequent **db2pd** gives us the results shown in Example 3-7.

*Example 3-7 db2pd output - HADR is in Peer state*

```
db2inst1@polonium:/usr/db2/backup> db2pd -d sample -hadr

Database Partition 0 -- Database SAMPLE -- Active -- Up 0 days 00:32:18

HADR Information:
Role      State              SyncMode HeartBeatsMissed  LogGapRunAvg (bytes)
Primary   Peer                  Sync       0                  0

ConnectStatus ConnectTime              Timeout
Connected    Thu Oct 19 10:03:44 2006 (1161277424) 3

LocalHost                LocalService
polonium                  DB2_HADR_2

RemoteHost                RemoteService      RemoteInstance
LEAD                      DB2_HADR_1         db2inst1

PrimaryFile PrimaryPg PrimaryLSN
S0000025.LOG 200      0x00000000072106AB

StandByFile StandByPg StandByLSN
S0000025.LOG 200      0x00000000072106AB
```

If you are managing HADR from the DB2 Control Center, you will have cataloged remote node and database aliases, so alternatively, you can issue takeover commands remotely by specifying the remote database alias and explicit DB2 instance user ID and password. For example, if we now want to switch the roles again so that database SAMPLE on server LEAD becomes the primary, we can issue the command from the POLONIUM server:

```
db2 takeover hadr on db sam_lead user db2inst1
```

**Tip:** There is no need to worry about revealing DB2 passwords, as you will be prompted for the password using an asterisk (\*) masked entry if you only specify the user.

If you have only cataloged the nodes and database alias from the primary server side when initially setting up HADR, there will not be any equivalent entries on the standby server. You can catalog them with the Control Center on the standby, or at the command line.

In summary, re-establishing (re-integrating) HADR should be as simple as restoring a backup from the primary over the top of the broken standby, and issuing appropriate start HADR commands, and no matter how long you take to do it, there is no need for downtime.





## HADR administration and monitoring

In this chapter we discuss the normal administration tasks performed in relation to a High Availability Disaster Recovery (HADR) system. We describe the methods of acquiring the status of HADR, including get snapshot, db2pd, and table functions new to DB2 Version 9.1. Finally, we present a section on reading DB2's db2diag.log file in relation to HADR.

## 4.1 Administering HADR systems

The focus of this section is to give you an insight on what is different on an HADR system as far as day-to-day DB2 administration tasks go. The administration of an HADR system is done mainly on the primary server, then replicated over to the standby. Regular database maintenance, such as backups, REORG, and RUNSTATS, are performed only on the primary. Logged operations are replicated to the standby, so it is important to understand what is a logged operation as compared to a non-logged operation.

### Replicated operations

The following operations are replicated to the standby from the primary:

- ▶ Data Definition Language (DDL):
  - Includes create and drop table, index, and more
- ▶ Data Manipulation Language (DML):
  - Insert, update, or delete
- ▶ Buffer pool:
  - Create and drop
- ▶ Table space:
  - Create, drop, or alter.
  - Container sizes, file types (raw device or file system), and paths must be identical on the primary and the standby.
  - Table space types (DMS or SMS) must be identical on both the servers.
  - If the database is enabled for automatic storage, then the storage paths must be identical.
- ▶ Online REORG:
  - Replicated. The standby's Peer state with the primary is rarely impacted by this, although it can impact the system because of the increase in the number of log records generated.
- ▶ Offline REORG:
  - Replicated, but depends on whether the REORG is re-clustering by an index:
    - Re-clustering REORG:

Operations are logged per hundreds or thousands of affected rows. The standby may fall behind in this case.

- Non-re-clustering REORG:

The entire operation is a single record generated after REORG on the primary. This operation has the greatest impact on the standby's ability to keep up with the primary.

- ▶ Stored procedures and user defined functions (UDF):
  - Creation statements are replicated.
  - HADR does *not* replicate the external code for stored procedures or UDF object library files. In order to keep these in synchronization between the primary and the standby, it is the user's responsibility to set these up on identical paths on both the primary and the standby servers. If the paths are not identical, invocation will fail on the standby.

## Non-replicated operations

The following database operations are not replicated by HADR:

- ▶ NOT LOGGED INITIALLY tables:
  - Tables created with the NOT LOGGED INITIALLY option specified are not replicated. The tables are invalid on the standby and after a takeover, attempts to access these tables will result in an error.
- ▶ BLOBs and CLOBs:
  - Non-logged Large OBjects (LOBs) are not replicated, but the space will be allocated and LOB data will be set to binary zeroes on the standby.
- ▶ Data links:
  - NOT SUPPORTED in HADR.
  - HADR fails when DATALINKS are set to YES
- ▶ Database configuration:
  - UPDATE DB CFG is *not* replicated.
  - Dynamic database configuration parameters can be updated to both the primary and the standby without interruption. For the database configuration parameters that need a recycle of the database we recommend you follow the rolling upgrade steps. See 7.1, "DB2 FixPak rolling upgrades" on page 166.
- ▶ Recovery history file:
  - NOT automatically shipped to standby.
  - You can place an initial copy of the history file by using a primary backup image to restore the history file to the standby using the RESTORE command with the REPLACE HISTORY FILE option for example:  

```
db2 restore database sample replace history file
```

- You can also update the history file on the standby from a primary backup image by the RESTORE command for example:

```
db2 restore database sample history file
```

- If a takeover operation occurs and the standby database has an up-to-date history file, backup and restore operations on the new primary will generate new records in the history file and blend seamlessly with the records generated on the original primary. If the history file is out-of-date or has missing entries, an automatic incremental restore might not be possible; instead, a manual incremental restore operation will be required.

We think that the recovery history file for the primary database will not be relevant to the standby database, as it contains DB2 recovery information specific to that primary server; DB2 keeps records of which backup file to use for recovery to prior points in time, among other things. In most HADR implementations, the standby system will only be used in the primary mode as a temporary measure, while the old primary server is being fixed, making the primary version of the recovery history file irrelevant; also, because of potentially different log file numbers and locations between the primary and the standby, incorrect for use on the standby server.

For more information, refer to the following source:

*DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228.

## Replicating LOAD operations

The following are LOAD operations that are logged on the primary and sent to the standby:

### ► LOAD COPY YES:

- Data is replicated on the standby if the standby can access the path or device specified in the load.
- Otherwise, the table space is marked bad and future log records for this table space are skipped.
- To avoid forced reintegration of the standby, you can recover the standby table space making sure that the path or devices are set correctly and re-run the LOAD command with the REPLACE option.
- If it is not possible to have both server accessing the same path or device, you can deactivate the standby database while the load operation is performed, perform the load on the primary, place a copy of the output file in the standby path, and then activate the standby database. For this situation, also see the use of registry variable DB2LOADREC in 6.2, “DB2 HADR registry variables” on page 143.

- ▶ **NONRECOVERABLE LOAD:**
  - The equivalent table on the standby will be marked bad and future log records regarding this table are skipped.
  - The standby database will skip future log records that pertain to this table. You can choose to re-issue the LOAD command on the primary with the COPY YES and REPLACE options specified to bring the table back, or you can drop the table to recover the space.
- ▶ **LOAD COPY NO:**
  - Load operations with the COPY NO option specified are not supported, so the command is automatically converted to LOAD NONRECOVERABLE on the primary.
  - In order for a LOAD COPY NO option to execute on the primary you can set the DB2\_LOAD\_COPY\_NO\_OVERRIDE registry variable to COPY YES and it converts your LOAD commands automatically into LOAD COPY YES. The DB2\_LOAD\_COPY\_NO\_OVERRIDE is ignored on the standby.

## Database backup

Database backup operations in relation to HADR are as follows:

- ▶ Backup on the standby is NOT allowed.
- ▶ Any of the standard database backups are allowed on the primary. If your goal is high availability, we recommend utilizing the online backup, in order to keep your database available.

## Database restore

Database restore operations in relation to HADR are as follows:

- ▶ Table space level restore on the primary or the standby is *not* allowed.
- ▶ Redirected restores are *not* allowed.
- ▶ A database restore can be performed on either the primary or the standby, but might reset the database to standard mode. In order to restore the database mode, the START HADR command with either the AS PRIMARY or AS STANDBY can be used.
- ▶ When restoring a primary database the old standby can be reintegrated by START HADR AS STANDBY, if the restore roll forward option is specified for a point in time at or later than the current log sequence number (LSN) on the standby.

## Log files

Database log file operations replicated in relation to archive logging are as follows:

- ▶ No replication of LOG ARCHIVES from the primary to the standby.
- ▶ Log archive occurs only on the primary to your specified archive method.
- ▶ The standby *does not* invoke USEREXIT.
  - It only extracts logs out of archive location for log replay.

## 4.2 Snapshot

The GET SNAPSHOT FOR DATABASE command is one way to get the status of the HADR databases. The snapshot can be used any time you need the current status of either the primary or the standby database.

db2 get snapshot for database on *database name*

The information returned represents the status of the database manager operations at the time the command is issued. HADR information is listed under the heading of HADR Status, as shown in Example 4-1.

*Example 4-1 Sample snapshot output*

---

HADR Status

Role	= Primary
State	= Peer
Synchronization mode	= Sync
Connection status	= Connected, 10/20/2006 15:17:12.954349
Heartbeats missed	= 0
Local host	= LEAD
Local service	= DB2_HADR_1
Remote host	= polonium
Remote service	= DB2_HADR_2
Remote instance	= db2inst1
timeout(seconds)	= 3
Primary log position(file, page, LSN) = S0000029.LOG, 532, 00000000082FC4C6	
Standby log position(file, page, LSN) = S0000029.LOG, 532, 00000000082FC4C6	
Log gap running average(bytes) = 0	

---

Within the HADR Status section, most of the output reflects your current settings set in the database configuration file. The settings that can have status changed are Role, State, Connection status, Heartbeats missed, Primary log position, Standby log position, and the Log gap running average.



## **Role**

This field indicates the database the snapshot is run against. The database role can be either Primary or Standby.

## **State**

The status for HADR *state* will depend on which server you are running the snapshot command. A description of each state is given here:

- ▶ Possible states on the primary database are as follows:
  - Peer:  
The primary is communicating with its standby and is shipping log buffer flushes.
  - Remote catchup:  
The primary is communicating with its standby and is shipping old logs.
  - Remote catchup pending:  
The primary is not shipping any logs (old or new) to the standby. The standby is trying to establish a connection with the primary. While in this state, you could potentially have a problem with the connection between the HADR servers.
- ▶ Possible states on the standby database are as follows:
  - Peer:  
Caught up to the tail of the log and is receiving currently-generated log data from the primary and replaying it.
  - Remote catchup:  
Not caught up to the tail of the log, has requested old log data from the primary that is not in the standby's log path and is not available through the standby's userexit.
  - Remote catchup pending:  
Finished its local catchup phase and is ready to receive log data from the primary. While in this state, you could potentially have a problem with the connection between the HADR servers.
  - Local catchup:  
Performing log replay from log data already on the disk of that system.

### ***Connection status***

The connection status returns the current connection status between the primary and the standby. This can be one of the following three possibilities:

- ▶ **Connected:**  
The database is connected to its partner node.
- ▶ **Disconnected:**  
The database is not connected to its partner node. If the disconnected status is not expected, you should check the status of the other HADR server either the primary or the standby, to determine if there is a database problem, or some network issue.
- ▶ **Congested:**  
The database is connected to its partner node, but the connection is congested. With a congested status, we suggest that you troubleshoot your network to pinpoint the problem for the congestion.

### ***Heartbeats missed***

Heartbeats are used by the HADR pair to check each other's status. If this number is zero, all heartbeats have been received. The higher this value the worse the connection condition is in. An HADR database expects at least one heartbeat message from the other database for each quarter of the time interval defined in the HADR\_TIMEOUT configuration parameter, or 30 seconds, whichever is shorter.

For example, if your HADR\_TIMEOUT is set to 40 seconds, the HADR database expects one heartbeat every 10 seconds. If no heartbeats are received in an HADR\_TIMEOUT interval, the partner is considered not responding and the connection is closed. In the case of a closed connection you must determine the cause; either the other HADR server is having problems or there are network problems. Once the problem is found and fixed, you can use the START HADR command to re-establish the HADR communication, if the logs are synchronized, otherwise you must re-establish HADR through a restore.

### ***Primary and standby log position***

These elements show the status on the log position of both the primary and the standby. Since log positions are transmitted from the partner node periodically, these may not be accurate. If log truncation takes place the “log gap running average” would not be accurate.

### ***Log gap running average***

This element shows the running average of the gap between the primary LSN and the standby LSN. The gap is measured in number of bytes. Since this is an average number, the value might not be 0 even the systems are in SYNC mode and in Peer state.

## 4.3 Monitoring HADR: db2pd

The current state of HADR can be gathered without explicit connection to a database using the **db2pd** command. Note that this command requires SYSADM authority in DB2, and also requires that the database be activated in order to get any useful output related to HADR.

Example 4-2 shows the command syntax and *monitor element* output for the **db2pd** command to get HADR information for the SAMPLE database on our HADR primary server (LEAD).

*Example 4-2 Output of db2pd for LEAD (primary)*

```
db2inst1@lead:~> db2pd -d sample -hadr
```

---

```
Database Partition 0 -- Database SAMPLE -- Active -- Up 1 days 01:52:39
```

HADR Information:

Role	State	SyncMode	HeartBeatsMissed	LogGapRunAvg (bytes)
Primary	Peer	Sync	0	0

ConnectStatus	ConnectTime	Timeout
Connected	Tue Oct 31 14:37:16 2006 (1162334236)	3

LocalHost	LocalService
LEAD	DB2_HADR_1

RemoteHost	RemoteService	RemoteInstance
polonium	DB2_HADR_2	db2inst1

PrimaryFile	PrimaryPg	PrimaryLSN
S0000031.LOG	289	0x00000000089D9F3B

StandByFile	StandByPg	StandByLSN
S0000031.LOG	289	0x00000000089D9F3B

---

Example 4-3 shows the output for the **db2pd** command to get HADR information for the SAMPLE database on our HADR standby server (POLONIUM).

*Example 4-3 Output of db2pd for POLONIUM (standby)*

---

```
db2inst1@polonium:~> db2pd -d sample -hadr
```

Database Partition 0 -- Database SAMPLE -- Active -- Up 1 days 01:54:36

HADR Information:

Role	State	SyncMode	HeartBeatsMissed	LogGapRunAvg (bytes)
Standby	Peer	Sync	0	0

ConnectStatus	ConnectTime	Timeout
Connected	Tue Oct 31 14:37:16 2006 (1162334236)	3

LocalHost	LocalService
polonium	DB2_HADR_2

RemoteHost	RemoteService	RemoteInstance
LEAD	DB2_HADR_1	db2inst1

PrimaryFile	PrimaryPg	PrimaryLSN
S0000031.LOG	289	0x00000000089D9F3B

StandByFile	StandByPg	StandByLSN
S0000031.LOG	289	0x00000000089D9F3B

---

As expected, the db2pd outputs on the HADR primary and the standby have differences and similarities. In our example (see Example 4-2 on page 101 and Example 4-3 above), the only noticeable differences are — in addition to LEAD having *HADR Role* of Primary and POLONIUM having Standby — that the *LocalHost*, *LocalService* monitor element values are swapped with *RemoteHost* and *RemoteService*.

In a long-running HADR pair, after numerous log writes and HADR stop/start activity, the primary and the standby log information (*PrimaryFile*, *StandByFile*, *PrimaryPg*, *StandbyPg*, *PrimaryLSN*, *StandbyLSN*) can differ. Having said that, in an *HADR State* of Peer, the output for each value shown in a **db2pd** command on the primary must still match the output for that same value shown in a **db2pd** command run on the standby at that same time. That is, if the db2pd output for StandByPg on the primary was 289, then it should also be 289 for the db2pd output for StandByPg on standby, even if PrimaryPg was some other value.

You may also notice that the *LogGapRunAvg* value is often not 0 bytes, even when *HADR State* is Peer.

The *log gap running average* value actually represents the average of the difference between the LSN of the primary and the standby, at the last time the primary and the standby databases were communicating. If a primary log is truncated for any reason, a gap will appear as the primary has moved to the start of the next log and the standby is still at the end of the last log. This gap should close after the next log write by the primary.

For more information about db2pd and HADR monitoring, refer to the following sources:

- ▶ *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228
- ▶ *DB2 9.1 Command Reference*, SC10-4226
- ▶ *System Monitor Guide and Reference*, SC10-4251

## 4.4 Monitoring HADR: Administrative view and table function

With DB2 9, a new way to view the various *monitor elements* within HADR is to use either an *administrative view*, or a *table function*.

*Monitor elements* are the individual attributes for various functional areas inside DB2 which are monitored. That is, they have counters or point-in-time status values updated as relevant events occur. Related monitor elements are referred to as *logical data groups*.

For the HADR logical data group in DB2 9, administrative views and table functions allow applications and scripts alike to use simple SQL queries and get a single value for a single HADR monitor element (or formatted output with multiple columns as required) rather than attempting to write complicated string manipulation routines or use **awk** and **grep**, to strip the output of **db2pd** or **db2 get snapshot for database**.

Example 4-4 and Example 4-5 show some sample syntax to effectively get the same result, specifically, the current value of the HADR\_STATE monitor element.

*Example 4-4 SQL table function to get the current primary HADR\_STATE*

---

```
> db2 "select hadr_state from table (SNAP_GET_HADR('sample',-1)) as SH"
```

```
HADR_STATE
-----
PEER
```

```
1 record(s) selected.
```

---

*Example 4-5 SQL Administrative View to get the current primary HADR\_STATE*

---

```
> db2 "select HADR_STATE from sysibmadm.snaphadr"
```

```
HADR_STATE
-----
PEER
```

```
1 record(s) selected.
```

---

Note that these are SQL queries and require a prior connection to the database. This means that with current versions of HADR, you are unable to use either administrative views or table functions to get the values of the monitor elements for a standby database. Attempts to run these commands on the standby will result in:

SQL1024N A database connection does not exist. SQLSTATE=08003

As you can guess, attempts to connect to a standby database result in:

SQL1776N The command cannot be issued on an HADR standby database.  
Reason code = "1".

The available monitor elements for the HADR logical data group are as shown in Example 4-6. A full description of each monitor element can be found in *DB2 9.1 Administrative SQL Routines and Views*, SC10-4293.

*Example 4-6 Output of describe table SYSIBMADM.SNAPHADR*

---

Column name	Type schema	Type name	Length	Scale	Null
-----					
SNAPSHOT_TIMESTAMP	SYSIBM	TIMESTAMP	10	0	Yes
DB_NAME	SYSIBM	VARCHAR	128	0	Yes
HADR_ROLE	SYSIBM	VARCHAR	10	0	Yes
HADR_STATE	SYSIBM	VARCHAR	14	0	Yes

---

HADR_SYNCMODE	SYSIBM	VARCHAR	10	0	Yes
HADR_CONNECT_STATUS	SYSIBM	VARCHAR	12	0	Yes
HADR_CONNECT_TIME	SYSIBM	TIMESTAMP	10	0	Yes
HADR_HEARTBEAT	SYSIBM	INTEGER	4	0	Yes
HADR_LOCAL_HOST	SYSIBM	VARCHAR	255	0	Yes
HADR_LOCAL_SERVICE	SYSIBM	VARCHAR	40	0	Yes
HADR_REMOTE_HOST	SYSIBM	VARCHAR	255	0	Yes
HADR_REMOTE_SERVICE	SYSIBM	VARCHAR	40	0	Yes
HADR_REMOTE_INSTANCE	SYSIBM	VARCHAR	128	0	Yes
HADR_TIMEOUT	SYSIBM	BIGINT	8	0	Yes
HADR_PRIMARY_LOG_FILE	SYSIBM	VARCHAR	255	0	Yes
HADR_PRIMARY_LOG_PAGE	SYSIBM	BIGINT	8	0	Yes
HADR_PRIMARY_LOG_LSN	SYSIBM	BIGINT	8	0	Yes
HADR_STANDBY_LOG_FILE	SYSIBM	VARCHAR	255	0	Yes
HADR_STANDBY_LOG_PAGE	SYSIBM	BIGINT	8	0	Yes
HADR_STANDBY_LOG_LSN	SYSIBM	BIGINT	8	0	Yes
HADR_LOG_GAP	SYSIBM	BIGINT	8	0	Yes
DBPARTITIONNUM	SYSIBM	SMALLINT	2	0	Yes

The benefit of the administrative view is the simplicity of the SQL SELECT statement interface, in addition to being available for viewing from the DB2 Control Center. Figure 4-1 shows most of the column names available in the SNAPHADR administrative view, which is essentially a Control Center equivalent of a DESCRIBE TABLE SQL statement.

This was achieved by clicking the **SAMPLE** database, which we have set up in HADR mode, connecting to the database through the **Connect** option in the right pane, clicking **Views** in the left hand tree, and finding and clicking the **SNAPHADR** view (SYSIBMADM schema). Double-clicking the **SNAPHADR** view would get us a single row result, with the current values for each of the HADR monitor elements.

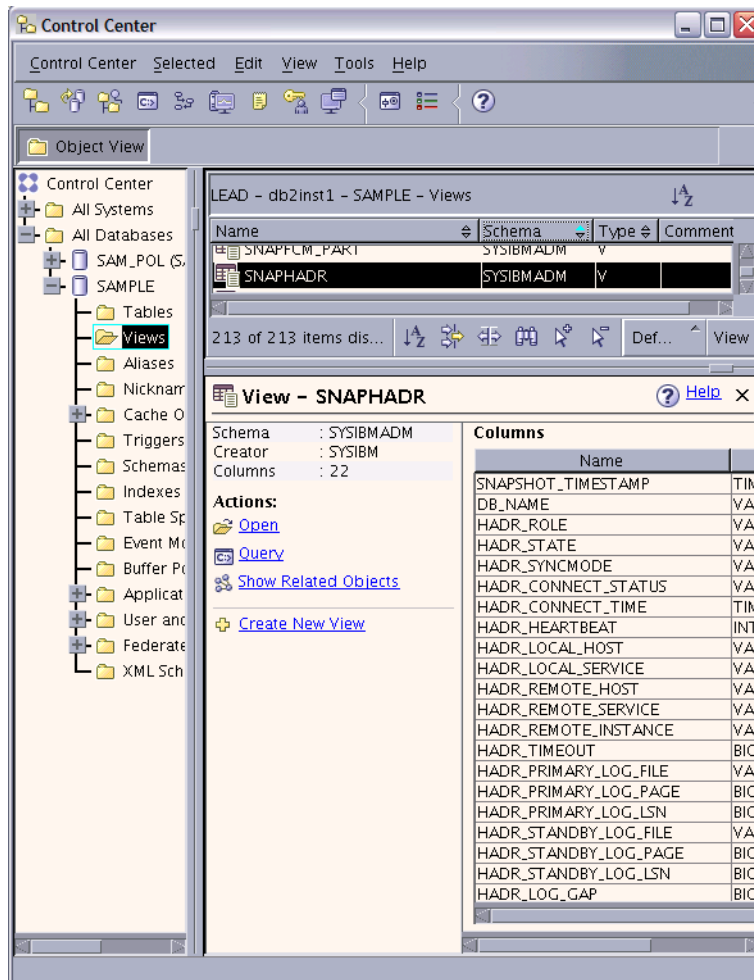


Figure 4-1 SNAPHADR administrative view as seen through DB2 Control Center

For more details about administrative views and table function, refer to the following DB2 manuals:

- ▶ *DB2 9.1 Administrative SQL Routines and Views*, SC10-4293
- ▶ *DB2 9.1 System Monitor Guide and Reference*, SC10-4251



## 4.5 Monitoring HADR: db2diag

It is not only possible to get certain basic status information about HADR from the db2diag.log, but depending on what you need, it can be very efficient to simply awk or grep the tail of the db2diag.log file. For example, from db2diag.log, you can easily find out whether the primary server thinks it is currently in the Peer state or not. In addition, using the **db2diag** system command allows you to filter and format the output of the db2diag.log file. This neither requires overhead to achieve a connection to the DB2 database, nor interface through any API to get to the DB2 snapshot or monitor element data.

Example 4-7 and Example 4-8 show the various states of progression through HADR startup to achieve the Peer state, and the syntax of the HADR messages you will see. This can be useful whether you are just manually eyeballing the output, or if you are scanning through the db2diag.log with some string searching algorithm for specific text.

We have provided these examples specifically to show what happens in parallel time lines on the primary and the standby after an initial HADR reintegration. The db2diag.log text shown in Example 4-7 was from just after the backup file had been restored on the standby, and HADR was started, first on the standby server (POLONIUM), and then on the primary server (LEAD).

We show the events occurring on the standby first in Example 4-7, as this is where events actually commenced first.

### *Example 4-7 Progression of HADR status from startup to Peer state on the standby*

---

```
2006-10-31-14.36.55.969031-480 I563982G415      LEVEL: Warning
PID       : 13964                TID  : 2988689072  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000          DB   : SAMPLE
APPHDL    : 0-976              APPID: *LOCAL.db2inst1.061031223655
AUTHID    : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-10-31-14.36.55.979794-480 E564398G310      LEVEL: Event
PID       : 20290                TID  : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE   : HADR state set to None (was None)

2006-10-31-14.36.55.986878-480 I564709G333      LEVEL: Warning
PID       : 20290                TID  : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20100
MESSAGE : HADR_LOCAL_HOST polonium mapped to 9.43.86.69

2006-10-31-14.36.55.987731-480 I565043G330      LEVEL: Warning
PID       : 20290                TID  : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20110
MESSAGE : HADR_REMOTE_HOST LEAD mapped to 9.43.86.68
```

```

2006-10-31-14.36.55.988115-480 I565374G299      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHosts, probe:20157
MESSAGE : HADR is using IPv4.

2006-10-31-14.36.56.010573-480 E565674G312      LEVEL: Event
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-Boot (was None)

2006-10-31-14.36.56.010899-480 I565987G315      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster, probe:21251
MESSAGE : Info: Replaymaster Starting...

2006-10-31-14.36.56.012106-480 I566303G345      LEVEL: Warning
PID      : 13934          TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE : Info: HADR Startup has begun.

2006-10-31-14.36.56.012967-480 I566649G332      LEVEL: Warning
PID      : 13934          TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:300
MESSAGE : Starting Replay Master on standby.

2006-10-31-14.36.56.013245-480 I566982G317      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrStartReplayMaster, probe:21252
MESSAGE : Info: Replaymaster request done.

2006-10-31-14.36.56.013480-480 E567300G322      LEVEL: Event
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-LocalCatchup (was S-Boot)

2006-10-31-14.36.56.013664-480 I567623G294      LEVEL: Warning
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduS, probe:20341
MESSAGE : Info: Standby Started.

2006-10-31-14.36.56.013899-480 I567918G419      LEVEL: Warning
PID      : 13964          TID : 2988689072  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000          DB : SAMPLE
APPHDL   : 0-976          APPID: *LOCAL.db2inst1.061031223655
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21152
MESSAGE : Info: HADR Startup has completed.

2006-10-31-14.36.56.020021-480 I568338G402      LEVEL: Severe
PID      : 20290          TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1      NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
MESSAGE : Failed to connect to primary. rc:
DATA #1 : Hexdump, 4 bytes
0xBF968D8 : 1900 0F81                      ....

2006-10-31-14.36.56.020398-480 I568741G346      LEVEL: Severe

```

```

PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduAcceptEvent, probe:20280
RETCODE  : ZRC=0x810F0019=-2129723367=SQL0_CONN_REFUSED "Connection refused"

2006-10-31-14.36.56.014136-480 E569088G348      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000      DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:920
MESSAGE  : ADM1602W Rollforward recovery has been initiated.

2006-10-31-14.36.56.315699-480 E569437G391      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000      DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpReplayMaster, probe:1740
MESSAGE  : ADM1603I DB2 is invoking the forward phase of the database
           rollforward recovery.

2006-10-31-14.36.56.316070-480 I569829G464      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000      DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpForwardRecovery, probe:710
DATA #1 : <preformatted>
Invoking database rollforward forward recovery,
lowtranlsn 00000000088B800C in log file number 31
minbufflsn 00000000088B800C in log file number 31

2006-10-31-14.36.56.332364-480 I570294G374      LEVEL: Warning
PID      : 13934                TID : 2988689072  PROC : db2agnti (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000      DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlprecm, probe:2000
DATA #1 : <preformatted>
Using parallel recovery with 3 agents 4 QSets 12 queues and 2 chunks

2006-10-31-14.36.56.338124-480 I570669G323      LEVEL: Info
PID      : 20284                TID : 2988689072  PROC : db2lfr (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, data protection services, sqlpgPostLogMgrToRetrieve, probe:1050
DATA #1 : <preformatted>
RTStatus is in state 16 at index 0.

2006-10-31-14.36.56.338917-480 I570993G322      LEVEL: Warning
PID      : 14923                TID : 2988689072  PROC : db2shred (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000      DB   : SAMPLE
APPHDL   : 0-975
FUNCTION: DB2 UDB, recovery manager, sqlpshrEdu, probe:18300
MESSAGE  : Maxing hdrLCUEndLsnRequested

2006-10-31-14.36.56.423126-480 E571316G338      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-RemoteCatchupPending (was S-LocalCatchup)

2006-10-31-14.37.16.046636-480 E571655G346      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to S-RemoteCatchupPending (was S-RemoteCatchupPending)

2006-10-31-14.37.16.066181-480 E572002G339      LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000

```

```

FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-RemoteCatchup (was S-RemoteCatchupPending)

2006-10-31-14.37.16.066431-480 I572342G312          LEVEL: Warning
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSPrepareLogWrite, probe:10260
MESSAGE : RCUStartLsn 000000000888800C

2006-10-31-14.37.16.692591-480 E572655G329          LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-NearlyPeer (was S-RemoteCatchup)

2006-10-31-14.37.16.801856-480 E572985G320          LEVEL: Event
PID      : 20290                TID : 2988689072  PROC : db2hadrs (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE : HADR state set to S-Peer (was S-NearlyPeer)

```

---

To sum up the states of progression on the standby:

- ▶ *None*: HADR has not yet started, the state is set to None.
- ▶ *S-Boot*: This is the initial phase.
- ▶ *S-LocalCatchup*: This indicates that any data from the standby's log path sitting in the log replay queue is being processed.
- ▶ *S-RemoteCatchupPending*: HADR has reached the end of the local logs in the replay queue, and is now waiting for a connection to the primary in order to see if there are further logs yet to be sent across for replay.
- ▶ *S-RemoteCatchup*: HADR has made a connection to the primary, and commences receiving active/archive logs from the primary log path and replaying those logs in order to get closer to the Peer state (as opposed to Peer state, where logs are sourced from the memory on the primary).
- ▶ *S-NearlyPeer*: HADR has finished replaying all logs on the standby local disk, and is waiting for the primary to suspend log writing, and complete reading any primary on-disk logs for processing on the primary.
- ▶ *S-Peer*: From the HADR standby's point of view, it has achieved the Peer state with the primary, according to the rules of the current SYNCMODE. The source of logs in the Peer state is the primary log buffer. That is, in-memory logs, as opposed to the primary on-disk source of logs in the case of RemoteCatchup.

These states which an HADR standby database goes through, from HADR START and database activation through to achieving the Peer state, are also described in Chapter 7 of the *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228. The output in the db2diag.log file describes internal states. There is also a prefix to show whether the state is for a primary (P-) or for a standby (S-) database. Correspondingly, this summary of states for the standby can be applied to the same states in Example 4-8 for the primary.

Defining the difference between the internal and the external states can be simplified as meaning the difference between the following states:

- ▶ The state of HADR at a lower, more accurate internal level, updated at a very high rate inside the db2diag.log file, with indicators of the primary and standby role in the state value
- ▶ The states displayed by the snapshot, db2pd, and table functions

While the rendition of external states is less granular, it is still effective for the purpose of monitoring HADR, and the `hdrMonitorState()` API function is able to map internal to external equivalent state.

Example 4-8 shows the equivalent actions occurring on the primary in the same parallel time line.

*Example 4-8 Progression of HADR status from Startup to Peer state on the primary*

---

```

2006-10-31-14.37.15.358606-480 I471198G393      LEVEL: Warning
PID       : 6585                TID  : 2988889776  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
APPHDL    : 0-125              APPID: *LOCAL.db2inst1.061031231208
AUTHID    : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21151
MESSAGE   : Info: HADR Startup has begun.

2006-10-31-14.37.15.362854-480 E471592G310      LEVEL: Event
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE    : HADR state set to None (was None)

2006-10-31-14.37.15.372100-480 I471903G329      LEVEL: Warning
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20100
MESSAGE   : HADR_LOCAL_HOST LEAD mapped to 9.43.86.68

2006-10-31-14.37.15.372600-480 I472233G334      LEVEL: Warning
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHostNamesToIp, probe:20110
MESSAGE   : HADR_REMOTE_HOST polonium mapped to 9.43.86.69

2006-10-31-14.37.15.372779-480 I472568G299      LEVEL: Warning
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrResolveHosts, probe:20157
MESSAGE   : HADR is using IPv4.

2006-10-31-14.37.15.395413-480 E472868G312      LEVEL: Event
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE    : HADR state set to P-Boot (was None)

2006-10-31-14.37.15.397848-480 I473181G294      LEVEL: Warning
PID       : 29494                TID  : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1             NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20301
MESSAGE   : Info: Primary Started.

```

```

2006-10-31-14.37.16.504640-480 E473476G330      LEVEL: Event
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-RemoteCatchupPending (was P-Boot)

2006-10-31-14.37.16.513132-480 I473807G397      LEVEL: Warning
PID      : 6585              TID : 2988889776  PROC : db2agent (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
APPHDL   : 0-125             APPID: *LOCAL.db2inst1.061031231208
AUTHID   : DB2INST1
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduStartup, probe:21152
MESSAGE  : Info: HADR Startup has completed.

2006-10-31-14.37.17.133766-480 E474205G339      LEVEL: Event
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)

2006-10-31-14.37.17.144621-480 I474545G313      LEVEL: Warning
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrEduP, probe:20445
MESSAGE  : remote catchup starts at 00000000088B800C

2006-10-31-14.37.17.146188-480 I474859G330      LEVEL: Warning
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrTransitionPtoNPeer, probe:10645
MESSAGE  : near peer catchup starts at 00000000088B800C

2006-10-31-14.37.17.250094-480 E475190G329      LEVEL: Event
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-NearlyPeer (was P-RemoteCatchup)

2006-10-31-14.37.17.258634-480 E475520G320      LEVEL: Event
PID      : 29494              TID : 2988889776  PROC : db2hadrp (SAMPLE) 0
INSTANCE: db2inst1           NODE : 000
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState, probe:10000
CHANGE  : HADR state set to P-Peer (was P-NearlyPeer)

```

---

## Filtering db2diag.log output

Using the unfiltered output of db2diag.log, the command shown in Example 4-9 is used to see if the primary is still in the Peer state or not. This is especially useful for the standby or cluster manager to know before issuing any **db2 takeover hadr on db ... by force** command.

*Example 4-9 Status checker script to be run continually on the HADR primary*

---

```

tail -0 -f db2diag.log | awk '
/CHANGE \: HADR state set to P-RemoteCatchupPending \(was P-Peer\)
{system("'"$WRITE_STATUS"' 1")}
/CHANGE \: HADR state set to P-Peer \(was P-NearlyPeer\)
{system("'"$WRITE_STATUS"' 0")}

```

---

Example 4-10 shows an example of using db2diag to filter the output of db2diag.log, with accompanying output as seen from our LEAD server.

*Example 4-10 db2diag filtering and timestamping specific output from the db2diag.log file*

---

```
db2diag -gi "funcname:=hdr" -fmt
"%{tsmonth}~%{tsday}~%{tshour}~%{tsmin}~%{tssec} %{process} %{changeevent}\t
%{message}"

10-11-16.58.17 db2agent (SAMPLE) 0 Info: HADR Startup has begun.
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR state set to None (was None)
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR_LOCAL_HOST LEAD mapped to 9.43.86.68
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR_REMOTE_HOST polonium mapped to 9.43.86.69
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR is using IPv4.
10-11-16.58.17 db2hadrp (SAMPLE) 0 HADR state set to P-Boot (was None)
10-11-16.58.17 db2hadrp (SAMPLE) 0 Info: Primary Started.
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchupPending (was P-Boot)
10-11-16.58.18 db2agent (SAMPLE) 0 Info: HADR Startup has completed.
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)
10-11-16.58.18 db2hadrp (SAMPLE) 0 remote catchup starts at 000000000659000C
10-11-16.58.18 db2hadrp (SAMPLE) 0 near peer catchup starts at 000000000659000C
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-NearlyPeer (was P-RemoteCatchup)
10-11-16.58.18 db2hadrp (SAMPLE) 0 HADR state set to P-Peer (was P-NearlyPeer)
10-12-09.20.18 db2hadrp (SAMPLE) 0 Zero bytes received. Remote end may have closed connection
10-12-09.20.18 db2hadrp (SAMPLE) 0 HADR primary rcv error:
10-12-09.20.18 db2hadrp (SAMPLE) 0
10-12-09.20.18 db2hadrp (SAMPLE) 0
10-12-09.20.18 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchupPending (was P-Peer)
10-12-09.20.59 db2hadrp (SAMPLE) 0 HADR state set to P-RemoteCatchup (was P-RemoteCatchupPending)
```

---

For more details of monitoring HADR, refer to *DB2 9.1 Data Recovery and High Availability Guide and Reference*, SC10-4228.

## 4.6 Monitoring summary

These sections on get snapshot, db2pd, administrative views and table functions, and db2diag.log sum up the methods by which you can keep track of the HADR monitor elements, outside of the GUI interfaces to those same methods, such as the Manage HADR dialog box in the DB2 Control Center.

As you can appreciate, each method has a situation in which it will be best suited:

- The get snapshot and db2pd output are similar enough to be used interchangeably, with some minor considerations. Although the get snapshot output contains all database monitor elements rather than just the HADR and as such can fill a few screens with output, it only requires SYSMON as a minimum authorization, whereas db2pd requires SYSADM.

- ▶ Administrative views and table functions with their extremely simple interface and specifically granular output, would be perfectly suited to a custom-built monitoring application for an operator flight deck. This could sit running in the background without having to run a dedicated DB2 Control Center and all the associated memory overhead. However, both these methods require a database connection and as such do not currently work on standby databases.
- ▶ Monitoring using db2diag.log requires no database connection overhead and is therefore extremely high performance oriented, it will give HADR state output for both the primary and the standby sides. Unfortunately, it requires a fair bit of string manipulation logic to get the information you need, it does not contain all the monitor element information, and is a historical point-in-time event log, not an on-demand resource. You cannot ask for the current HADR state and have it refreshed into the db2diag.log for retrieval. Monitoring must occur on a continual basis.





## Automatic client reroute

In this chapter we describe an important DB2 feature known as *automatic client reroute* (ACR). ACR enables a DB2 UDB client application to recover from a loss of communications so that the application can continue its work with minimal interruption.

We discuss the following topics:

- ▶ Automatic client reroute overview
- ▶ Automatic client reroute tuning
- ▶ Limitations
- ▶ Examples

## 5.1 Automatic client reroute overview

When there is a loss of communication between the client and the database server, the client receives a communication failure that terminates the application with an error. If high availability is important, you should implement a redundant setup and the ability to failover to the redundant setup. Automatic client reroute is a DB2 feature that enables a DB2 Client to recover from a loss of connection to the DB2 server by rerouting the connection to an alternate server. This automatic connection rerouting occurs without any manual intervention.

### 5.1.1 DB2 automatic client reroute with HADR

Figure 5-1 illustrates the automatic client reroute with HADR.

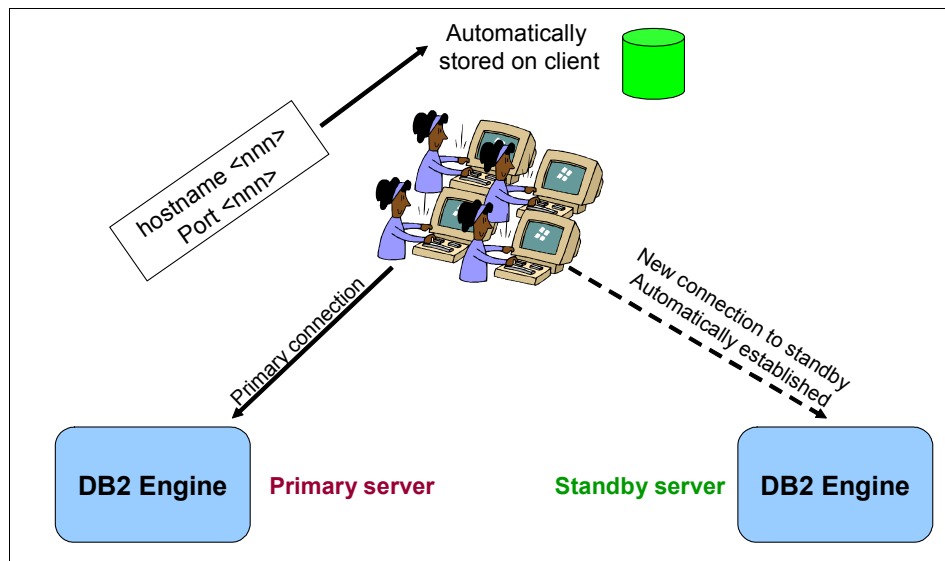


Figure 5-1 Automatic client reroute implementation in HADR

To enable the automatic client reroute feature, the DB2 server is configured with the name of an alternative location that the client can access. The `UPDATE ALTERNATE SERVER FOR DATABASE` command is used to define the alternate server location on a particular database.

```
db2 update alternate server for database db2 using hostname XYZ port  
YYYY
```

The alternate host name and port number is given as part of the command. The location is stored in the system database directory file at the server instance.

The alternate server location information is propagated to the client when the client makes a connection to the DB2 server. If communication between the client and the server is lost for any reason, DB2 Client will attempt to re-establish the connection by using the alternate server information.

The automatic client reroute feature could be used within the following configurable environments:

- ▶ Enterprise Server Edition (ESE) with the data partitioning feature (DPF)
- ▶ SQL Replication
- ▶ Tivoli System Automation (TSA)
- ▶ High Availability Cluster Multiprocessor (HACMP)
- ▶ High Availability Disaster Recovery (HADR)

### 5.1.2 DB2 automatic client reroute in action

If an alternate server is specified, automatic client reroute is activated when a communication error (SQLCode -30081) or a SQLCode -1224 is detected. These error codes are not returned to the client but they activate DB2 automatic client reroute feature. In a high availability disaster recovery (HADR) environment, the automatic client reroute will also be activated if SQLCode -1776 is returned.

When automatic client reroute is activated, DB2 Client code first attempts to re-establish the connection to the original server, if that fails, it will try the alternate server. The DB2 Client will continue the attempt to connect with the original server and the alternate server, alternating the attempts between the two servers until it gets a successful connection for ten minutes from the moment it detected a communication failure expire whichever happens first. The timing of these attempts varies from the initial very rapid connection attempts between the two servers with a gradual lengthening of the intervals between the attempts. This is the default behavior. But we can control the interval durations using registry variables.

*Table 5-1 Automatic client reroute connection intervals*

Time	Interval between attempts
0 to 30 seconds	0 seconds
30 to 60 seconds	2 seconds
1min to 2 minutes	5 seconds
2min to 5 minutes	10 seconds
5min to 10minutes	30 seconds

Once a connection is successful, the SQLCode -30108 or SQLCode -4498 (for Java Universal Driver JCC clients) is returned to indicate that a database connection has been re-established following the communication failure. Example 5-1 shows the SQLCode 4498.

*Example 5-1 SQLCode 4498*

---

```
SQLException: com.ibm.db2.jcc.c.ClientRerouteException:  
[ibm][db2][jcc][t4][2027][11212] A connection failed but has been  
re-established. The host name or IP address is "9.43.86.111" and the  
service name or port number is 50,030.Special registers may or may not  
be re-attempted (Reason code = 1).SQLCode: -4498
```

---

When the application receives this message, the state of the transactions that were executing is "unknown." Because the client did not get a reply from the server, it does not know if the transaction succeeded or not. The transaction may have been committed on the old server, and it may also have been replicated to the new server.

It is the application's responsibility to check if the transaction has succeeded or not and resubmit if needed. The application can continue with the next transaction or the application can have logic in place to trap this exception and check the status of the transactions and retry only the failed transaction to make the application seamless to the communication failure. The client code only returns the error for the original communications failure (SQL30081N) to the application if the re-establishment of the client communications is not possible to either the original or alternate server after 10 minutes.

Figure 5-2 shows the sequence of steps when there is a connection failure to the primary database.

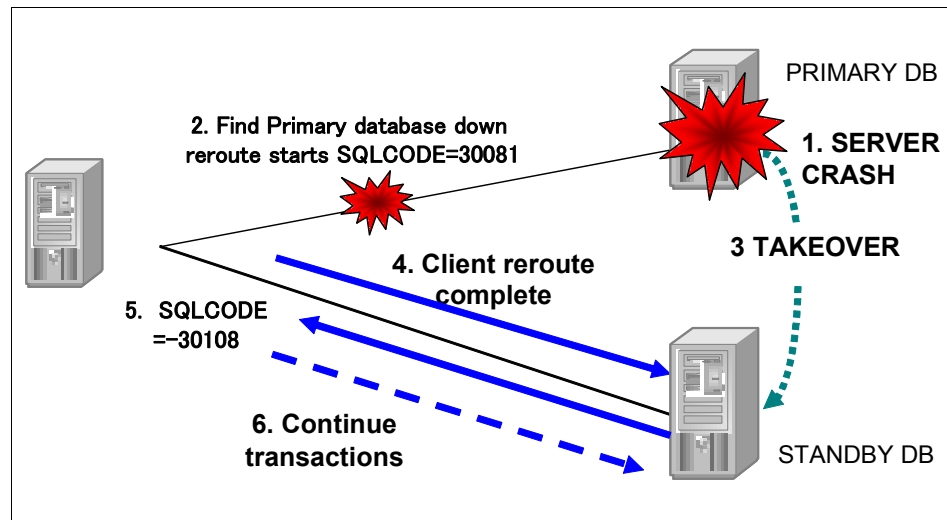


Figure 5-2 Automatic client reroute in action

## 5.2 Automatic client reroute tuning

DB2 Client actually tries to connect to the original server first before connecting to the alternate server when client reroute is triggered. Therefore, to enable fast reroute to the alternate server, the connection to the original server which has just died has to get a timeout error quickly. There are three options to achieve this:

- **Tuning TCP/IP network:**

To make the automatic client reroute perform better, we can tune the TCP/IP network parameters of the operating system on which DB2 application is running. For example, on AIX operating system, tune `tcp_keepinit`. The default value 150 means that 75 seconds has to be passed before the connection is really routed to the alternate server.

- **Configuring DB2 registry variables:**

The operating system network parameters affects all the applications on the system. So it is recommended to configure the following DB2 client reroute registry variables which affect only DB2 applications. You need to tune the parameters which will enable DB2 to quickly figure out that the network connection is not alive.

- **DB2TCP\_CLIENT\_CONTIMEOUT:**

This registry variable specifies the number of seconds a client waits for the completion on a TCP/IP connect operation. If a connection is not established in the seconds specified, then the DB2 database manager returns the error code -30081. There is no timeout if the registry variable is not set or is set to 0.

- **DB2TCP\_CLIENT\_RCVTIMEOUT:**

This registry variable specifies the number of seconds a client waits for data on a TCP/IP receive operation. If data from the server is not received in the seconds specified, then the DB2 database manager returns the error code -30081. There is no timeout if the registry variable is not set or is set to 0. This variable should be set to longer than the longest query execution time. Otherwise, the client will timeout during a long query and think that the connection is broken.

By default, the automatic client reroute feature retries the connection to a database repeatedly for up to 10 minutes. It is, however, possible to configure the exact retry behavior using one or both of these two registry variables:

- **DB2\_MAX\_CLIENT\_CONNRETRIES:**

This registry variable specifies the maximum number of connection retries attempted by the automatic client reroute. If this registry variable is not set, but the **DB2\_CONNRETRIES\_INTERVAL** is set, the default value is 10.

- **DB2\_CONNRETRIES\_INTERVAL:**

This registry variable specifies the sleep time between consecutive connection retries, in number of seconds. If this variable is not set, but **DB2\_MAX\_CLIENT\_CONNRETRIES** is set, the default value of **DB2\_CONNRETRIES\_INTERVAL** is 30 seconds.

If neither **DB2\_MAX\_CLIENT\_CONNRETRIES** nor **DB2\_CONNRETRIES\_INTERVAL** is set, the automatic client reroute feature reverts to its default behavior.

Automatic client reroute tries to connect to both original and alternate servers in a set of trials. In one set of trials, client connects four times as follows:

- Two times for original server
- Two times for alternate server

After one set of trial, client reroute waits for **DB2\_CONNRETRIES\_INTERVAL** and then repeats this trial process for **DB2\_MAX\_CLIENT\_CONNRETRIES**. After the trial reaches to **DB2\_MAX\_CLIENT\_CONNRETRIES**, it waits for **DB2\_CONNRETRIES\_INTERVAL** and then automatic client reroute fails with error code SQL30081.

For example, if the following registry variables had been set as follows:

- DB2\_MAX\_CLIENT\_CONNRETRIES=5
- DB2\_CONNRETRIES\_INTERVAL=3
- DB2TCP\_CLIENT\_CONTIMEOUT=10

DB2 automatic client reroute will take  $\{(10 \times 4) + 3\} \times 5 = 215$  seconds to give up and give a communications failure error SQL30081.

**Note:** Users of Type 4 connectivity with the DB2 Universal JDBC™ Driver should use the following two DataSource properties to configure the automatic client reroute:

- **maxRetriesForClientReroute:** Use this property to limit the number of retries if the primary connection to the server fails. This property is only used if the `retryIntervalClientReroute` property is also set.
- **retryIntervalForClientReroute:** Use this property to specify the amount of time (in seconds) to sleep before retrying again. This property is only used if the `maxRetriesForClientReroute` property is also set.

For CLI/ODBC, OLE DB, and ADO.NET applications, you can set a connection timeout value to specify the number of seconds that the client application waits for a reply when trying to establish a connection to a server before terminating the connection attempt and generating a communication timeout.

If client reroute is activated, you need to set the connection timeout value to a value that is equal to or greater than the maximum time it takes to connect to the server. Otherwise, the connection might timeout and be rerouted to the alternate server by client reroute. For example, if on a normal day it takes about 10 seconds to connect to the server, and on a busy day it takes about 20 seconds, the connection timeout value should be set to at least 20 seconds.

In the `db2clini.ini` file, you can set *ConnectTimeout* CLI/ODBC configuration keyword to specify the time in seconds to wait for a reply when trying to establish a connection to a server before terminating the attempt and generating a communication timeout. By default the client waits indefinitely for a reply from the server when trying to establish a connection.

If *ConnectTimeout* is set and client reroute is activated, a connection will be attempted only once to the original server and once to the alternate server. Since the *ConnectTimeout* value is used when attempting to connect to each server, the maximum waiting time will be approximately double the specified value for *ConnectTimeout*. If neither server can be reached within the time limit specified by the keyword, an error message will be received, as shown in Example 5-2.

#### *Example 5-2 Communication error message*

---

```
SQL30081N  A communication error has been detected.  Communication
protocol being used: "TCP/IP".  Communication API being used:
"SOCKETS".  Location where the error was detected: "<ip
address>".Communication function detecting the error: "<failing
function>".  Protocol specific error code(s): "<error code>", "**",
"*.  SQLSTATE=08001
```

---

When making DB2 database connections through the DB2.NET Provider, you can set the value of `ConnectTimeout` property of `DB2Connection` when passing the `ConnectionString`, as shown in Example 5-3.

#### *Example 5-3 Set ConnectionString for DB2 .NET Provider*

---

```
String connectionString =
"Server=srv:5000;Database=test;UID=adm;PWD=abd;Connect Timeout=30";
DB2Connection conn = new DB2Connection(connectionString);
conn.Open();
```

---

In this example, if the connection attempt takes more than thirty seconds, the connection attempt will be terminated. The setting of `ConnectionString` has higher priority than the registry variables or `db2cli.ini` setting.

Similarly, for OLE .NET Data Provider and ODBC .NET Data Provider, you can also set `ConnectTimeout` property by `ConnectionString` (Example 5-4).

#### *Example 5-4 Set ConnectionString for ODBC .NET Data Provider*

---

```
OleDbConnection con = new
OleDbConnection("Provider=IBMDADB2.DB2;Data
Source=TEST;ConnectTimeout=15");

OdbcConnection con = new
OdbcConnection("DSN=test;ConnectTimeout=15");
```

---

The setting of `ConnectionString` has higher priority than the settings in the registry variables or in the `db2cli.ini` file.

- Using the same cluster manager:

The third option to have fast failover is to set up an HADR pair where the primary and standby databases are serviced by the same cluster manager. Refer to example in 5.4.3, “Example 3: Involving an HADR database in HACMP cluster” on page 127 for more details. In that example, we define a service IP address for the connection from DB2 clients, and take it over when the primary fails. We use the service IP address for both “catalog tcpip node” executed on the client and “update alternate server” executed on the server. This method is the fastest but it requires an HA cluster configuration.



## 5.3 Limitations

There are some limitations when using the automatic client reroute feature:

- ▶ TCP/IP is the only supported communications protocol by automatic client reroute in DB2 UDB servers. Even if DB2 is setup for a loopback, TCP/IP communications protocol must be used in order to accommodate the automatic client reroute feature.
- ▶ Automatic client reroute feature is supported only in DB2 for Linux, UNIX and Windows operating systems in both the client and the server side. Other DB2 families do not currently support this feature.
- ▶ The DB2 server installed in the alternate host server must be the same version (but could have a higher FixPak) when compared to the DB2 installed on the original host server.
- ▶ If the connection is re-established to the alternate server, any new connection to the same database alias will be connected to the alternate server. If you want any new connection to be established to the original location in case the problem on the original location is fixed, there are three methods to achieve it:
  - You need to take the alternate server offline and allow the connections to fail back over to the original server. This requires that the original server has been cataloged using the `UPDATE ALTERNATE SERVER` command such that it is set to be the alternate location as the alternate server.
  - You could catalog a new database alias to be used by the new connections.
  - You could uncatalog the database entry and re-catalog it again.
- ▶ The alternate server information is always kept in memory. If you do not have the authority to update the database directory (or because it is a read-only database directory), other applications will not be able to determine and use the alternate server, because the memory is not shared among applications.
- ▶ The client will be unable to reestablish the database connection if the alternate location has a different authentication type than the original location. The same authentication is applied to all alternate locations.
- ▶ When there is a communication failure, all session resources such as global temporary tables, identity, sequences, cursors, server options (`SET SERVER OPTION`) for federated processing and special registers are all lost. The application is responsible to re-establish the session resources in order to continue processing the work. You do not have to run any of the special register statements after the connection is re-established, because DB2 will replay the special register statements that were issued before the communication error.

However, some of the special registers will not be replayed. They are:

- SET ENCRYPTPW
- SET EVENT MONITOR STATE
- SET SESSION AUTHORIZATION
- SET TRANSFORM GROUP

**Note:** If the client is using CLI, JCC Type 2 or Type 4 drivers, after the connection is re-established, for those SQL statements that have been prepared against the original server, they are implicitly prepared again with the new server. However, for embedded SQL routines (for example, SQC or SQX applications), they will not be prepared again.

- The Universal DB2 driver is showing an error that should be reported as a `StaleConnectionException` in application servers like WebSphere but it is currently not mapped that way. The error is for a DB2 HADR client failover exception, which will cause error codes -4498 or -30108 to be returned. These exceptions should be mapped to `StaleConnectionException`. Mapping these codes to `StaleConnectionException` will simplify application retry logic and enable the ability to use the application server connection pool purge policy of “entire pool” to more quickly establish new connections.

**Note:** At the time of writing this book, the fix for the WebSphere APAR PK06078 was targeted for inclusion in fixpack 5.0.2.12 and 5.1.1.6. Refer to the Recommended Updates page for delivery information:

<http://www.ibm.com/support/docview.wss?rs=180&uid=swg27004980>

## 5.4 Examples

Automatic client reroute is activated by updating the alternate server for database on the DB2 server. You can use `DB2 LIST DATABASE DIRECTORY` command to confirm the alternate server as shown in Example 5-5.

*Example 5-5 Check the alternate server setting*

---

### **db2 list db directory**

```
System Database Directory
Number of entries in the directory = 1
Database 1 entry:
Database alias           = SAMPLE
Database name            = SAMPLE
Local database directory = /home/hadrinst/dbdir
Database release level   = a.00
```

Comment	=
Directory entry type	= Indirect
Catalog database partition number	= 0
Alternate server hostname	= 9.43.86.111
Alternate server port number	= 50030

---

The client applications will receive SQLCode -30108 to indicate that the connection has been re-established. The transaction that was executing when the communications failure occurred will be in an unknown state. The client application has to be designed such that it can ignore this error and proceed to the next transaction or it could retry this transaction after confirming that it has not succeeded.

Let us see how automatic client reroute works in the following cases:

- ▶ Automatic client reroute involving a non-HADR database
- ▶ Automatic client reroute involving an HADR database
- ▶ Automatic client reroute involving an HADR database in HACMP cluster

### 5.4.1 Example 1: Involving a non-HADR database

In this example, we use two DB2 host systems KANAGA and ATLANTIC. At KANAGA, a database SAMPLE is created. Furthermore, the database SAMPLE is also created at the alternate server ATLANTIC with port number 50030.

At the client machine, the database SAMPLE is cataloged at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at server KANAGA as follows:

```
db2 update alternate server for database sample using hostname atlantic  
port 50030
```

Without having the automatic client reroute feature set up, if there is a communication error when executing a transaction, the application receives an error SQL30081N. With the automatic client reroute feature set up, DB2 will try to establish the connection to host KANAGA again. If it is still not working, DB2 will try the alternate server location (host ATLANTIC with port 50030). Assuming there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or resubmit the transaction if it has failed. Since we do not have HADR in this case, one has to consider how to keep the two databases in sync and the standby to be online when the primary is down.

From the client machine, if we connect to the database SAMPLE and run a SELECT query as in Example 5-6, we see that the queries are run at KANAGA.

*Example 5-6 Run queries at primary DB2 system*

---

```
$db2 select id from staff fetch first 1 row only
ID
-----
    10

    1 record(s) selected.

$ db2 "select substr(host_name,1,8) from table(db_partitions()) as t"
1
-----
kanaga.i

    1 record(s) selected.
```

---

If there is a communications error while executing the same query to select one row from the STAFF table, the automatic client reroute will route the query to the alternate server. See Example 5-7.

*Example 5-7 Connection has been reroute*

---

```
$db2 select id from staff fetch first 1 row only
SQL30108N A connection failed but has been reestablished.
The hostname or IP address is
"192.168.1.20" and the service name or port number
is "50030". Special registers may or may not be reattempted
(Reason code = "1"). SQLSTATE=08506

$db2 select id from staff fetch first 1 row only
ID
-----
    10

    1 record(s) selected.

$ db2 "select substr(host_name,1,8) from table (db_partitions()) as t"
1
-----
atlantic.i

    1 record(s) selected.
```

---

## 5.4.2 Example 2: Involving an HADR database

In this example, we set up HADR configuration between KANAGA and ATLANTIC for database SAMPLE. At KANAGA, a primary database SAMPLE is created. A standby HADR database is also created at host ATLANTIC with port 50030.

At the client machine, a database SAMPLE is cataloged at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at KANAGA as follows:

```
db2 update alternate server for database sample using hostname atlantic  
port 50030
```

To enable an alternate server for the standby server at ATLANTIC which can fail back to primary server at KANAGA port 50030 which can be reintegrated into the HADR pair, you need to run this command on host ATLANTIC.

```
db2 update alternate server for database sample using hostname kanaga  
port 50030
```

Without having the automatic client reroute feature set up, if there is a communication error, the application will receive an error SQL30081N. When the primary database server fails, applications will receive an error SQL30081N even if HADR successfully brings up the standby server.

If the automatic client reroute feature is set up, DB2 will try to establish the connection to host KANAGA again. If it is still not working, DB2 will try the alternate server location (host ATLANTIC with port 50030). Assuming there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or retry only the failed transactions. The alternate server location is where the HADR standby is located. For client reroute to succeed, HADR has to successfully bring up the standby server as the new primary. This process has to be initiated through the HADR takeover command either manually or by automating this takeover through some other HA software like HACMP.

## 5.4.3 Example 3: Involving an HADR database in HACMP cluster

In this scenario, we set up an HADR pair on KANAGA and ATLANTIC where the primary and standby databases are serviced by the same cluster manager. HADR is established between KANAGA and ATLANTIC for database SAMPLE. At the server KANAGA, primary database SAMPLE is created. A standby HADR database is also created at host ATLANTIC with port 50030.

HACMP software will detect when the active node is down and do the IP takeover as well as issue the HADR takeover command to make the standby as the new primary server. In this case the HACMP service IP address is configured as 9.43.86.111.

To activate automatic client reroute, you need to update the alternate server for database SAMPLE at KANAGA using the service IP as the host name as follows:

```
db2 update alternate server for database sample using hostname  
9.43.86.111 port 50030
```

To enable an alternate server for the standby server ATLANTIC which can fail back to primary server at KANAGA port 50030, you need to run the following command on host ATLANTIC. Again, the service IP is used as the host name.

```
db2 update alternate server for database sample using hostname  
9.43.86.111 port 50030
```

After failback to old primary server, the old primary server (KANAGA) should be reintegrated into the HADR pair.

At the client machine, the database SAMPLE is cataloged at service IP address 9.43.86.111 at port 50030.

In this HADR with HACMP environment, service IP address (9.43.86.111) was used as the alternate server on both the primary and the standby servers. The service IP address (9.43.86.111) was also used as the host name when cataloging the database SAMPLE at the client. If the primary database server at KANAGA goes down, DB2 will keep trying to establish the connection to host service IP address 9.43.86.111.

Initially the service IP owner is KANAGA. Once the HACMP detects that the primary node KANAGA is down, it performs takeover and the standby node ATLANTIC owns the resource. DB2 continues trying to connect to the service IP at alternate server location (host 9.43.86.111 with port 50030). Assuming that HACMP has successfully done the IP takeover and moved the resource group to ATLANTIC and there is no communication error on the connection to the alternate server location, the application can then continue to run subsequent statements or retry the failed transactions.

In an HACMP/HADR cluster, we could also have configured the alternate servers similar to the “Example 2: Involving an HADR database” on page 127. This is another option to configure the alternate servers in an HACMP/HADR cluster.

At the client machine, catalog the database SAMPLE at node KANAGA. Node KANAGA references the host name KANAGA and port 50030.

At host KANAGA which is the primary HADR database, the alternate server is specified using the host name instead of service IP:

```
db2 update alternate server for database sample using hostname atlantic  
port 50030
```

At host ATLANTIC which is the standby HADR server, the alternate server name is also specified using host name instead of service IP:

```
db2 update alternate server for database sample using hostname kanaga  
port 50030
```

Using the service IP address as the alternate server configuration is faster as HACMP detects the node is down and does an IP fail over. When alternate client reroute detects a communications failure, it first tries to connect to the original server. There is a good chance that the HACMP IP takeover and resource group transition has succeeded and DB2 could get a successful connection on the original server address (service IP address 9.43.86.111) which is now transitioned to host ATLANTIC.

HACMP can make it all so seamless that sometimes when the client machine has cataloged the database at node using the Service IP address, it is not clear which host machine is currently the primary server. To find out the primary database server host name, you can execute the following query:

```
db2 select substr(host_name,1,8) from table (db_partitions()) as t
```

#### 5.4.4 Sample application code to handle automatic client reroute

In this section we show a pseudo code example of an embedded SQL C application and a Java application using DB2 Universal JDBC Type 2 or Type 4 drivers handling the Automatic client reroute exception.

##### Java application handling the ACR exception

After a connection is re-established, the DB2 Universal JDBC Driver throws a `java.sql.SQLException` to the application with `SQLCODE -4498`, to indicate to the application that the connection has been automatically re-established to the alternate server.

Example 5-8 shows the Java application code handling the automatic client reroute exception by catching the exception -4498.

*Example 5-8 Handling ACR exception in Java*

---

```
// In retry logic you don't have to retry connection or create
statement or preparestatement again

Connection con = DriverManager.getConnection(url, "USER", "PSWD");
Statement stmt = con.createStatement();
pstmt = con.prepareStatement("SELECT NAME FROM STAFF WHERE ID = ?");

// the statements above don't have to re-try when we get client
reroute exception

do { // while loop start
try {
// transaction logic
pstmt.setInt(1,10);
int rowsUpdated = stmt.executeUpdate(
    "UPDATE employee SET firstame = 'SHILI' WHERE empno = '10'");
con.commit();
}catch(SQLException se) { //deal with client reroute exception
    int errorcode = se.getErrorCode();
    System.out.println("SQLException: " + se);
    System.out.println("MySQLCode: " + errorcode);
    if(con != null){
    try{
        con.rollback();
    }catch(Exception e)
        { e.printStackTrace(); }
    } //endif
    if((errorcode == -30108) || (errorcode == -4498)){
        System.out.println("connection is re-established, re-executing
the failed transaction."); retry = true
    } finally { // close resources}
    } //end catch
    } while(retry)
```

---

### **Java application with JCC Type 4 driver**

Handling the ACR exception is identical in Type 2 and Type 4 JDBC drivers as given in Example 5-8. But implementing client reroute while using JCC Type 4 drivers requires a different setup when obtaining a connection to the database. JCC Type 2 drivers use a native client library specific to the data source to which they connect.



JCC type 4 driver is coded in pure Java, with no native client library component and implements the network protocol to a data source. JCC Type 4 clients do not require a DB2 installation on the client side server and can connect directly to the DataSource. Instead of allowing a client application to pick up connection information from the local database directory, the server name and port are explicitly included in the JCC connection attempt. If, however, at the time of connection, the server function has been taken over by an alternate server, the client will not only be unable to connect, but will not know where to find the alternate server information.

DB2 Universal JDBC Type 4 Driver client reroute support is available only for connections that use the *javax.sql.DataSource* interface. Example 5-9 shows an example of how we get a connection using a DataSource property file. DB2SimpleDataSource is one of the DB2 implementations of the DataSource interface.

---

*Example 5-9 Obtaining a database connection using DataSource interface*

---

```
DB2SimpleDataSource dataSource = new DB2SimpleDataSource();
    Properties prop = new Properties();
    FileInputStream dsPropFile = null;
    try{
        dsPropFile = new FileInputStream( DSname);
    }
    catch (FileNotFoundException fe)
    {
        System.out.println (fe.getMessage());
        throw fe;
    }
    prop.load( dsPropFile );
    dataSource.setServerName (prop.getProperty("serverName"));
    String portNum = prop.getProperty("portNumber");
    int portNo = (new Integer(portNum)).intValue() ;
    dataSource.setPortNumber (portNo);
    dataSource.setDatabaseName (prop.getProperty("databaseName"));
    dataSource.setUser (prop.getProperty("userName"));
    dataSource.setPassword (prop.getProperty("password"));
    Connection con=datasource.getConnection();
    con.setAutoCommit(false);
        Statement stmt = con.createStatement();
}
```

---

The best way to use a DataSource object is for your system administrator to create and manage it separately, using WebSphere or some other tool. The program that creates and manages a DataSource object also uses the Java Naming and Directory Interface™ (JNDI) to assign a logical name to the DataSource object. The JDBC application that uses the DataSource object can then refer to the object by its logical name, and does not need any information about the underlying data source. In addition, your system administrator can modify the data source attributes, and you do not need to change your application program.

To obtain a connection using a data source object:

1. From your system administrator, obtain the logical name of the data source to which you need to connect.
2. Create a Context object to use in the next step. The Context interface is part of the Java Naming and Directory Interface (JNDI), not JDBC.
3. In your application program, use JNDI to get the DataSource object that is associated with the logical data source name.
4. Use the DataSource.getConnection method to obtain the connection.

By using the javax.sql.DataSource interface, alternate server parameters can be picked up by the JCC application and kept in non-volatile storage on the client machine. The storage can be done using the JNDI API. If, for instance, a local file system is specified as the non-volatile storage, JNDI will create a .bindings file which will contain the required alternate server information.

After the current JVM™ is shut down, the information will then persist in that file until a new JVM is created. The new JVM will attempt to connect to the server. If the alternate server information has been updated, this will be updated on the client machine without requiring your intervention. If the server is missing however, the .binding file will be read and a new connection attempt will be made at the location of the alternate server. LDAP can also be used to provide non-volatile storage for the alternate server information. Using volatile storage is not recommended, as a client machine failure could result in the loss of alternate server data stored in memory.

To register the alternate server with JNDI and make it persistent:

1. Create an instance of DB2ClientRerouteServerList, and bind that instance to the JNDI registry. DB2ClientRerouteServerList is a serializable Java bean with four properties:
  - alternateServerName
  - alternatePortNumber
  - primaryServerName
  - primaryPortNumber

Getter and setter methods for accessing these properties are provided.

Example 5-10 shows the code sample of binding of an instance of `Db2ClientRerouteServerList` “address” to the JNDI registry.

*Example 5-10 Binding of Db2ClientRerouteServerList instance to JND*

---

```
// Create a starting context for naming operations
InitialContext registry = new InitialContext();

// Create a DB2ClientRerouteServerList object
DB2ClientRerouteServerList address = new
DB2ClientRerouteServerList();

// Set the port number and server name for the primary server
address.setPrimaryPortNumber(50030);
address.setPrimaryServerName("KANAGA.itso.ibm.com");

// Set the port number and server name for the alternate server
int[] port = {50030};
String[] server = {"ATLANTIC.itso.ibm.com"};
address.setAlternatePortNumber(port);
address.setAlternateServerName(server);
registry.rebind("serverList", address);
```

---

2. Assign the JNDI name of the `DB2ClientRerouteServerList` object to `DataSource` property `clientRerouteServerListJNDIName`. For example:

```
datasource.setClientRerouteServerListJNDIName("serverList");
```

If you use `DataSource` in WebSphere Application Server environment, this step is done in the administration console of WebSphere application Server.

Here is how the DB2 Universal JDBC Driver makes `DB2ClientRerouteServerList` persistent. After the database administrator specifies the alternate server location on a particular database at the server instance, the primary and alternate server locations are returned back to the client at connect time.

The DB2 Universal JDBC Driver creates an instance of referenceable object `DB2ClientRerouteServerList` and stores that instance in its transient memory. If communication is lost, the DB2 Universal JDBC Driver tries to re-establish the connection using the server information that is returned from the server. The *`clientRerouteServerListJNDIName`* `DataSource` property provides additional client reroute support at the disposal of the client.

The `clientRerouteServerListJNDIName` has two functions:

- ▶ Allows alternate server information to persist across JVMs.
- ▶ Provides an alternate server location in case the first connection to the database server fails.

The `clientRerouteServerListJNDIName` identifies a JNDI reference to a `DB2ClientRerouteServerList` instance in a JNDI repository for alternate server information. After a successful connection to the primary server, the alternate server information that is provided by `clientRerouteServerListJNDIName` is overwritten by the information from the server.

The DB2 Universal JDBC Driver will attempt to propagate the updated information to the JNDI store after a failover if `clientRerouteServerListJNDIName` property is defined. If `clientRerouteServerListJNDIName` is specified, primary server information specified in `DB2ClientRerouteServerList` will be used for connection. If a primary server is not specified, `serverName` information specified on the `DataSource` will be used.

A newly established failover connection is configured with the original `DataSource` properties, except for the server name and port number. In addition, any DB2 special registers that were modified during the original connection are re-established in the failover connection by DB2 Universal Driver JDBC Driver.

## Embedded SQL Program using C

After a connection is re-established, an embedded SQL C application will receive an exception `SQLCode -30108`, to indicate to the application that the connection has been automatically re-established to the alternate server.

Example 5-11 is a pseudo C code for a client application of automatic client reroute.

*Example 5-11 Handling the ACR exception in C program*

---

```
int checkpoint = 0;
check_sqlca(unsigned char *str, struct sqlca *sqlca)
{
    if (sqlca->sqlcode == -30081)
    { // as communication is lost, terminate the application right away
        exit(1);
    } else
    { // print out the error
        printf(...);
    }
    if (sqlca->sqlcode == -30108)
    { // connection is re-established, re-execute the failed transaction
        if (checkpoint == 0)
```

```

{ goto checkpoint0; }
else if (checkpoint == 1)
{ goto checkpoint1; }
else if (checkpoint == 2)
{ goto checkpoint2; }
....
exit;
}
}
}
main()
{
connect to mydb;
check_sqlca("connect failed", &sqlca);
checkpoint0:
EXEC SQL set current schema XXX;
check_sqlca("set current schema XXX failed", &sqlca);
EXEC SQL create table t1...;
check_sqlca("create table t1 failed", &sqlca);
EXEC SQL commit;
check_sqlca("commit failed", &sqlca);
if (sqlca.sqlcode == 0)
{ checkpoint = 1; }
checkpoint1:
EXEC SQL set current schema YYY;
check_sqlca("set current schema YYY failed", &sqlca);
EXEC SQL create table t2...;
check_sqlca("create table t2 failed", &sqlca);
EXEC SQL commit;
check_sqlca("commit failed", &sqlca);
if (sqlca.sqlcode == 0)
{ checkpoint = 2; }
...
}

```

---





## HADR best practice

In this chapter we describe the DB2 configuration parameters and registry variables that affect HADR. We also discuss the import factors to consider when implementing an HADR solution to obtain the most optimum HADR performance.

We cover the following topics:

- ▶ DB2 HADR configuration parameters
- ▶ DB2 HADR registry variables
- ▶ Recommendations and considerations
- ▶ Restrictions

## 6.1 DB2 HADR configuration parameters

Here we explore the relevant *database manager configuration parameters* (dbm cfg parm) and *database configuration parameters* (db cfg parm) that have some appreciable impact on HADR.

In most cases, there is no need to change these parameters manually after setting up HADR. However, it is always useful to know what to change and why, in the event that an architectural requirement is changed further down the track.

We do not provide specific performance recommendations in this section, as each environment will require tuning to match its own unique characteristics. Having said that, the information provided here, in the DB2 manual pages, and other references listed will be enough to give you a good idea of which settings will meet your requirements.

One thing to keep in mind, with configuration parameters, and the registry and environment variables in the next section, is that they can only be recognized and used while the database is in either *primary*, or *standby* role, and ignored otherwise. Still, it is important for those parameters/variables to be set and maintained on both servers of the HADR pair, for when the time comes to switch roles or perform a forced takeover.

### 6.1.1 Basic configuration parameters

The following terms can be found in DB2 9 Information Center (**Reference** → **Configuration Parameters** → **Database Systems**) and in *DB2 9 Performance Guide* SC10-4222: Appendix A. The same information is also given in *DB2 V8.2 Information Center and Administration Guide: Performance V8*, SC09-4821-01.

► **AUTORESTART:**

Consider setting this db cfg parm value to OFF when using automatic client reroute, so that a broken primary server does not come back online and restart in HADR primary role, causing a split-brain scenario.

In a non-HADR environment, it is good to leave AUTORESTART set to ON, as the database will automatically activate when the instance is started and the first application connection attempt is made, attempting log/crash recovery as required for any inflight/indoubt transactions that existed at the time the database was deactivated.

When AUTORESTART is left OFF, connection attempts receive an SQL1015N message.



► LOGINDEXBUILD:

This db cfg parm value should be set to ON so that maintenance operations on indexes, such as create, or REORG, on the primary database will be logged and carried out to match on the standby database.

► LOGRETAIN / LOGARCHMETHn:

You must have at least one db cfg parm set in order to use HADR. Circular logging is not supported.

LOGRETAIN is a legacy parameter from the days before a native archive logging method (LOGARCHMETHn) was supported. You can still use it to specify that DB2 active logs and archive logs should be stored together in one single location. The log file location is specified by LOGPATH and can be changed by updating NEWLOGPATH and deactivate/activate database commands.

LOGARCHMETHn (1,2) specifies the destination for the primary and mirror archive logs, respectively. This can be a disk, TSM, userexit, or vendor supplied destination driver. Having a value here means that after an active log is closed by DB2 (no more active transactions referring to it), it is moved out of the active log directory and sent to the specified target/destination.

► HADR\_DB\_ROLE:

This db cfg parm is configurable indirectly through HADR commands, to STOP, START, and TAKEOVER HADR. The possible values are:

– STANDARD:

The database is not in an HADR role and can be processed normally.

– PRIMARY:

The database is the HADR primary, all client interactions with the HADR pair occur here.

– STANDBY:

The database is the HADR standby, any client interaction attempts will receive an SQL1776N message. Only DB2's log replay engine is allowed to run against this database. You can issue the following command against an HADR standby database to see the log replay application:

```
db2 list applications all show detail
```

► HADR\_LOCAL\_HOST:

This db cfg parm refers to the host name or IP address for the current server. The reason for using a separate host field for HADR is to support the use of HADR specific network card, which is addressed by its own IP address or host name that is different from the “usual” name of the host.

► **HADR\_LOCAL\_SVC:**

Logically coupled with the above parameter, this db cfg parm specifies the port number or service name used for the local HADR component of the HADR database. Note that this is completely separate from the port/service assigned to a given DB2 Instance. The HADR service/port number is specific to each HADR database.

► **HADR\_REMOTE\_HOST:**

This db cfg parm specifies the host name or IP address of the remote server for the HADR paired databases.

► **HADR\_REMOTE\_INST:**

This db cfg parm specifies the DB2 Instance name on the remote server for the HADR paired databases. Note that there is no equivalent db cfg parm for the local server — it can be derived from the current instance environment variable.

► **HADR\_REMOTE\_SVC:**

Combined with the HADR\_REMOTE\_HOST and HADR\_REMOTE\_INST, this db cfg parm specifies the port number or service name used by the HADR component for the HADR database on the remote server. As with the HADR\_LOCAL\_SVC, this value is specific to the database, and any given pair of HADR databases must have two different TCPIP port numbers.

► **HADR\_SYNCMODE:**

This is an important db cfg parameter. The setting must match on both servers in the HADR pair.

The following values are possible:

– **SYNC:**

In this mode, log writes are considered successful only when logs have been written to log files on the primary database and when the primary database has received acknowledgement from the standby database that the logs have also been written to log files on the standby database. The log data is guaranteed to be stored on both servers.

– **NEARSYNC:**

In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and when the primary database has received acknowledgement from the standby system that the logs have also been written to main memory on the standby system. Loss of data occurs only if both sites fail simultaneously and if the target site has not been transferred to nonvolatile storage with all of the log data that it has received.

- ASYNC:

In this mode, log writes are considered successful only when the log records have been written to the log files on the primary database and have been delivered to the TCP layer of the primary system's host machine. Because the primary system does not wait for acknowledgement from the standby system, transactions might be considered committed when they are still on their way to the standby.

- ▶ HADR\_TIMEOUT:

This db cfg parm specifies the time in seconds which the DB2 HADR EDU will wait for any response from its HADR partner, before considering communication to have failed and closing connection. If HADR was in Peer state before closing connection, the primary would no longer follow Peer state semantics when HADR connection is closed. The response could be either heartbeat or acknowledgement (ACK) signals. Note that there is no timeout for ACK signal wait, and ACK signals are not even used for ASYNC mode. This value must also match on both sides of the HADR pair.

- ▶ INDEXREC:

Index recreation time is both a dbm and db cfg parm which specifies when, if necessary, DB2 will attempt to rebuild invalid indexes — specifically for HADR, whether this will occur during HADR log replay on the standby. The dbm cfg parm is meant to be used as the default for all databases for that instance, the db cfg parm, when not set to SYSTEM, will override that value. Possible values are:

- SYSTEM:

Applies to db cfg only; accept the value for INDEXREC in the dbm cfg.

- ACCESS:

Rebuild invalid index when the index is first accessed, and then rebuild on HADR standby during log replay.

- ACCESS\_NO\_REDO:

Rebuild invalid index when index is first accessed, but leave invalid on HADR standby, no rebuild during log replay. The index would be rebuilt after an HADR takeover and the first access of the underlying table.

- RESTART:

This is the default value. Rebuild of invalid index occurs after a RESTART DATABASE or HADR takeover on the normal or primary database, and on the standby database during log replay. The AUTORESTART db cfg parm effectively means that RESTART DATABASE is implicitly issued at the time an application attempts to connect to the database. Indexes are rebuilt asynchronously (no wait) at takeover time, and synchronously (wait) at restart time.

- RESTART\_NO\_REDO:

Rebuild of invalid index occurs after a RESTART DATABASE on the normal or primary database, but not on the standby during HADR replay. Rebuild would occur only after HADR takeover.

- SELF\_TUNING\_MEM:

From DB2 9 onwards, DB2 is able to dynamically assign memory for buffer pools, package cache, lock list, sort heap, and database shared memory, using this db cfg parm. Self Tuning Memory, even when set up on both servers, can only be active on an HADR primary database.

- LOGFILSZ:

This db cfg parm which specifies the size of active logs is taken from the primary and used by the standby so that the size of active log files on both servers match — the standby db cfg parm value of LOGFILSZ is ignored. This value is kept even after an HADR role switch or takeover, until such time as the database is restarted, when the local LOGFILSZ value is used for active log file size.

## 6.1.2 Automatic client reroute configuration parameters

This minute subset of DB2 configuration parameters is strictly speaking, not part of HADR, or even, strictly speaking, a configuration parameter, but it is used by ACR.

The command **db2 update alternate server for database** sets values stored in the system database directory file for the server and DB2 Instance port number. As an example for database SAMPLE, where our current primary server is named LEAD, and the alternate would be named POLONIUM, both using TCP/IP port 50001 for the DB2 instance DB2INST1:

```
db2 update alternate server for database sample using hostname  
POLONIUM port 50001
```

The mechanism of automatic client reroute, and how it is a separate entity to HADR, are explained in Chapter 5, “Automatic client reroute” on page 115.

## 6.2 DB2 HADR registry variables

This section examines the DB2 registry and operating system shell environment variables pertaining to HADR.

The following can be found in DB2 9 Information Center (**Reference** → **Configuration Parameters** → **Database Systems**) and in *DB2 9 Performance Guide*, SC10-4222: Appendix B. The information are also equivalent in *DB2 V8.2 Information Center and Administration Guide: Performance V8*, SC09-4821-01.

► **DB2\_HADR\_BUF\_SIZE:**

This registry variable is only recognized while the database is in the standby role. By default, it is the size of the HADR standby log receive buffer. The value of this variable is calculated as twice the value of LOGBUFSZ.

The standby replay mechanism retrieves logs directly from the log receive buffer. If the standby is slow in replaying logs, and the primary keeps sending more logs to the standby, the log receive buffer will eventually become “full”, preventing the standby from receiving more logs. Saturation of the receive buffer will cause transactions on the primary to be blocked until receive buffer has more room to receive log pages.

If the synchronization mode is ASYNC, the network pipeline from the primary to the standby will eventually fill up and the primary will not be able to send any more logs. This is called “congestion”. In ASYNC mode, network congestion and saturation of the standby's receive buffer will stall the primary log processing. Hence transactions on the primary are to be blocked until congestion is cleared, provided the standby can send heartbeat and the primary can receive the standby's heartbeat.

If the synchronization mode is SYNC or NEARSYNC, the primary is likely to be able to send out one more batch of logs after the standby receive buffer is full. This batch of logs will be buffered in the network pipeline between the primary and the standby. In SYNC and NEARSYNC modes, although the primary is not likely to encounter congestion when sending logs, it will still have to wait for acknowledgement messages when the standby buffer is full. Thus the primary transaction processing will eventually be blocked in all the synchronization modes when the standby receive buffer is full.

A larger standby receive buffer size (set by registry variable DB2\_HADR\_BUF\_SIZE) will allow the standby to absorb load peaks, reducing the chance of slowing the primary down. But if the sustained throughput of the standby log processing is lower than the primary log creation rate, a larger buffer will still be filled up and the primary be slowed down.

If your primary load is uneven, consider a larger standby receive buffer to absorb the peaks. Consider that in certain scenarios, both SYNC and NEARSYNC modes see a significant benefit from a very generous increase of the DB2\_HADR\_BUF\_SIZE variable, whereas the manual might give the impression that only ASYNC mode would benefit.

We recommend that you perform basic transactional throughput testing using a larger DB2\_HADR\_BUF\_SIZE value on each of the three synchronization modes to determine the best solution for your own environment. NEARSYNC is the default mode, and in many cases will provide a solution which is closest in performance to not running HADR.

► **LOAD\_COPY\_NO\_OVERRIDE**

This variable is not specific to HADR. However it can have either an adverse or beneficial effect on an HADR database depending on the value you assign to it. The variable is ignored on the standby databases, but applied on either a primary or a standard role database.

By default, when a **db2 load...copy no** command is run on an HADR primary database, DB2 will convert that to NONRECOVERABLE, that table space will be placed in a copy pending (read only) state, and the HADR standby will cease to match the primary. The table on the standby will be marked bad, and logs from the primary will not be applied there. This state can be corrected by subsequent execution of a **db2 load...copy yes** command.

If this registry variable is set to COPY YES (with a valid copy destination as part of the syntax), the load is converted such that it will automatically be used with log replay on the HADR standby to maintain data integrity. Note that the load command would still have to be using a valid target which the HADR standby database can access, in order for this to be effective (i.e. a shared drive with matching path, or a mutually accessible TSM target).

► **DB2LOADREC**

This registry variable can be used when the **db2 load...copy yes** backup image location on the HADR standby does not match the target location it was sent to from the HADR primary. This can happen when shared disk relative mapping does not match on both servers, or a method of file transfer outside DB2's control is occurring. Essentially, the value of DB2LOADREC is set to a plain text file which contains all required media information about the load copy backup image. The structure and required content of this file is fully described in the *DB2 v9 Data Recovery and High Availability Guide and Reference*, from page 165, and also in *DB2 9 Data Movement Utilities Guide and Reference*, SC10-4227, page 131.

As of DB2 9, there are no Operating System Shell environment variables used to influence the operation of HADR.

## 6.3 Recommendations and considerations

In this section we point out the important things to consider when implementing an HADR solution. There is no one perfect way of configuring HADR; everything depends on the user's requirements. If transaction performance is critical, then you might have to sacrifice HADR takeover time. If there is high logging activity and the network cannot keep up with the logging, you might have to sacrifice the transaction performance. You have to weigh the pros and cons of each of the following factors on the user requirements when tuning the HADR.

### 6.3.1 DB2 transaction performance

HADR setup could have a very slight impact on the performance of the DB2 transactions. With a stricter log shipping mode, the potential performance impact is higher. The tuning in an HADR setup is just the usual tuning of DB2 parameters in a non-HADR environment. There are no specific HADR parameters to be tuned. Each environment will require tuning to match its own unique characteristics.

Depending on the log shipping mode, the performance of the in-flight transactions on the primary can get affected when the standby goes down.

The primary database continuously polls the network. If the network or the standby machine goes down, the primary database will detect the condition as soon as the primary host machine operating system detects the condition. If the standby database crashes, but the standby machine and the network are still up, in most cases, the primary machine and therefore the primary database can detect the failure and close the connection quickly. Closing the connection will cause the primary database to change from Peer state to disconnected state, unblocking pending transactions. HADR state changes from peer to disconnected.

If the primary machine is unable to report the standby or network failure in time, DB2 will rely on HADR\_TIMEOUT to disconnect. If the primary does not receive any message from the standby for HADR\_TIMEOUT seconds (default 120 seconds) since the last heartbeat, it will change the state to disconnected.

With SYNC and NEAR SYNC log shipping mode, a commit statement for in-flight transactions does not complete until HADR state is changed to disconnected. This is because for SYNC and NEAR SYNC modes, the primary will have to wait for the acknowledgement. With ASYNC log shipping mode, in-flight transactions do not suffer from a network disconnection or the standby down and commit statement for them complete immediately. This is because in ASYNC mode when the primary sends logs, the operating system network layer might return send success, but buffer the data in the network path, or it might return "congestion".

For ASYNC mode, if the primary keeps sending more logs, it will eventually cause congestion. If send returns congestion, the primary transaction processing will be blocked immediately in ASYNC mode also. When there is a congestion or the receive buffer is full, the standby knows that it cannot receive heartbeat from primary anymore. It updates its heartbeat receive time and continues to send heartbeat to primary, so the standby will not drop connection after HADR\_TIMEOUT.

On the primary side, since the primary is receiving heartbeat from the standby, the primary also will not drop connection after HADR\_TIMEOUT. Since the standby can send heartbeat and the primary can receive the standby's heartbeat, the HADR pair will stay connected. In the worst case, unresponsive standby or network could cause the primary transactions to be blocked as long as the primary can receive heartbeat from the standby.

### **6.3.2 DB2 utilities performance**

Most DB2 utilities have no significant performance difference in the HADR environment. CPU usage also does not change significantly with HADR. But the DB2 REORG (both INPLACE/SHADOW mode) shows a performance slowdown. Table reorganization performance degrades with stricter log shipping mode.

### **6.3.3 How to reduce the takeover time**

Customers would like to keep HADR takeover time to a minimum. By reducing the amount of transaction rollback after takeover on the standby, you can open up the database to the clients sooner.

There is not a significant difference in takeover time between the different log shipping modes. When primary fails, forced takeover is usually faster than unforced. But if the standby detects the failure and is already disconnected, unforced takeover cannot be issued because the standby is no longer in Peer state.

If you know for sure that the primary has failed and would like to failover, issue forced takeover on the standby. If the standby has not detected the failure and is still in Peer state, it will allow unforced takeover. But because the primary is down, unforced takeover will not get any response from the primary and will hang for HADR\_TIMEOUT period. So if you have determined that primary is down, you should issue forced takeover directly.



### 6.3.4 Seamless takeover

HADR does not automatically detect a failed primary and issue a takeover. This process is manual. Once the user has determined the primary is down, the user can issue the takeover command. But manual takeover might not be acceptable in certain situations. In that case, you can configure HADR with a clustering software such as HACMP or TSA or MSCS to make this takeover completely seamless. The cluster software detects that the primary is down and issues an HADR takeover. Refer to Chapter 10, “HADR with clustering software” on page 253 for more details on implementing this solution.

### 6.3.5 Applications with high logging rate

For applications that have a very high logging activity, you have to make sure that the network is capable of transmitting that load of data. The standby database should also be powerful enough to replay the logged operations of the database as fast as they are generated on the primary. We recommend identical primary and standby hardware.

DB2 HADR does not currently support compression of the log files before sending them to the standby in order to improve this situation. However, depending on the synchronization mode that you are using, the logging rate on the primary can be reduced to the rate of transfer on the network by choosing a more strict form of log shipping mode, such as SYNC mode, which forces the primary to wait for the standby to acknowledge that it has received the log record before it can proceed. The network bandwidth is the limit of logging rate.

In most systems, the logging capability is not driven to its limit. Some systems do not see much difference among the three synchronization modes, or with the HADR enabled or disabled. This behavior is usually seen in systems where logging is not the bottleneck of database performance.

### 6.3.6 Performance implications of HADR\_TIMEOUT

If one database does not receive any message from the other one for the HADR\_TIMEOUT period, the connection is closed.

Try to avoid setting the HADR\_TIMEOUT for too long. In Peer state, if the standby is not responding, transactions on the primary can hang for HADR\_TIMEOUT period.

Setting the HADR\_TIMEOUT too short can also impact the performance. You get a false alarm on the connection. Frequent disconnecting and reconnecting are a waste of resources. HA protection also suffers as disconnection brings the primary out of Peer state. We recommend setting it to at least 10 seconds.

### 6.3.7 Network considerations

The primary database ships the log records to the standby database server over TCP/IP network when the primary does a log flush. In the primary database, SQL agent EDUs produce logs and write the logs into the database log buffer, whose size is controlled by the database configuration parameter LOGBUFSZ. The loggw EDU consumes the logs by writing them to disk. The write operation is called log flushing. Each write operation (and the logs written) is called a flush. For HADR primary databases, each flush is also sent to the standby database. For each write call, there is a matching send call that delivers the exact block of log data to the TCP layer.

The logger does not wait for the log buffer to be full to flush it. A transaction commit will generate a flush request. If there is no request, loggw will still wake up from time to time to flush the log buffer. The size of each log flush is non-deterministic. When there are multiple client sessions, multiple requests can be combined. The logger is designed to be self tuning. If a flush takes a longer time, when the logger completes the flush, there will be more outstanding requests, and therefore a stronger grouping effect on commit requests, improving performance by reducing the number of writes.

If there is only one client session, each commit will cause a log flush. If the synchronization mode is SYNC or NEARSYNC and the network is not very fast, the round trip messaging required by the synchronization mode can have a great impact on performance.

If the primary log buffer is large, each flush is likely to be large too. The send request to TCP can involve large blocks of data. The TCP layer should be tuned to handle such requests efficiently.

For HADR to work efficiently, a high speed, high capacity network between the primary and standby database is highly recommended. This will enable the standby to receive and acknowledge the logs as quickly as possible. We also recommend that the bandwidth of the network link be greater than the bandwidth of the logs generated. The network is vital to the performance of HADR, so it is important to tune the network. For better performance in an HADR environment, the network between both databases should be set up correctly. In terms of AIX TCP tuning, set these two OS parameters that affect the network performance:

► tcp\_recvspace:

The tcp\_recvspace tunable specifies how many bytes of data the receiving system can buffer in the kernel on the receiving sockets queue. The tcp\_recvspace tunable is also used by the TCP protocol to set the TCP window size, which the TCP uses to limit how many bytes of data it will send to the receiver to ensure that the receiver has enough space to buffer the data. The tcp\_recvspace tunable is a key parameter for TCP performance

because TCP must be able to transmit multiple packets into the network to ensure that the network pipeline is full. If TCP cannot keep enough packets in the pipeline, then performance suffers. You can set the `tcp_recvspace` tunable by `no -o tcp_recvspace=[value]` command. You can also set interface-specific values of `tcp_recvspace` from `smit chinnet` menu.

► `tcp_sendspace`:

The `tcp_sendspace` tunable specifies how many bytes of data the sending application can buffer in the kernel before the application is blocked on a send call. The TCP-socket send buffer is used to buffer the application data before it is sent to the receiver by the TCP protocol. The default size of the send buffer is basically specified by the `tcp_sendspace` tunable value. You should set the `tcp_sendspace` tunable value at least as large as the `tcp_recvspace` value, and for higher speed adapters, the `tcp_sendspace` value should be at least twice the size of the `tcp_recvspace` value.

For more details on TCP tuning parameters, see the following Web site:

[http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/tcp\\_streaming\\_workload\\_tuning.htm](http://publib.boulder.ibm.com/infocenter/pseries/v5r3/index.jsp?topic=/com.ibm.aix.prftungd/doc/prftungd/tcp_streaming_workload_tuning.htm)

## Network delays

Network latency affects transaction performance in only SYNC and NEARSYNC modes. The slowdown in system performance as a result of using SYNC mode can be significantly larger than that of the other synchronization modes. In SYNC mode, the primary database sends log pages to the standby database only after the log pages have been successfully written to the primary database log disk.

In order to protect the integrity of the system, the primary database waits for an acknowledgement from the standby before notifying an application that a transaction was prepared or committed. The standby database sends the acknowledgement only after it writes the received log pages to the standby database disk. The resulting overhead is the log write on the standby database plus round-trip messaging.

In NEARSYNC mode, the primary database writes and sends log pages in parallel. The primary then waits for an acknowledgement from the standby. The standby database acknowledges as soon as the log pages are received into its memory. On a fast network, the overhead to the primary database is minimal. The acknowledgement might have already arrived by the time the primary database finishes local log write.

For ASYNC mode, the log write and send are also in parallel; however, in this mode the primary database does not wait for an acknowledgement from the standby. Therefore, network delay is not an issue. Performance overhead is even smaller with ASYNC mode than with NEARSYNC mode.

## Network down

The primary database is continuously polling the network. If the network goes down, the primary database will detect the condition as soon as the primary host machine operating system detects the condition. If the primary machine and therefore the primary database can detect the failure, it will close the connection quickly. Closing the connection will cause the primary database to change from Peer state to disconnected state, unblocking pending transactions.

If the primary machine is unable to report network failure timely, DB2 will rely on HADR\_TIMEOUT to disconnect. If the primary does not receive any message from the standby for HADR\_TIMEOUT seconds, it will disconnect. In such a scenario, transactions on the primary will be blocked while the primary waits on the timeout. In ASYNC mode, when the primary sends the logs, these could be buffered in the network path and delay the blocking of transactions on the primary. But as the primary keeps sending more logs, it will eventually cause congestion and the transactions will be blocked.

When there is congestion, the standby knows that it cannot receive heartbeat from the primary anymore and updates its heartbeat receive time and continues to send heartbeat to the primary, so the standby will not drop connection after HADR\_TIMEOUT. On the primary side, since the primary is receiving heartbeat from the standby, the primary will not drop connection either, after the HADR\_TIMEOUT. Since the standby can send heartbeat and the primary can receive standby's heartbeat, the HADR pair will stay connected. In the worst case, an unresponsive network will cause primary transactions to be blocked until the primary does not receive any heartbeat or message from the standby for HADR\_TIMEOUT seconds. If no heartbeat is received by the primary from the standby until HADR\_TIMEOUT, the HADR state changes to DISCONNECTED.

When the primary state changes to DISCONNECTED, the primary gives up the transfer of the log records to the standby database. The primary database continues processing transactions even though it cannot transfer logs to the standby database. There is a possibility of data loss, if you execute a TAKEOVER BY FORCE command when there is a data gap between the primary database and the standby database. The primary will continue to listen on the HADR TCP/IP port waiting for the standby to come online. When the network recovers, the standby database tries to catch up to the primary until they return to peer status.

Figure 6-1 shows what happens when the network is down between the primary and secondary.

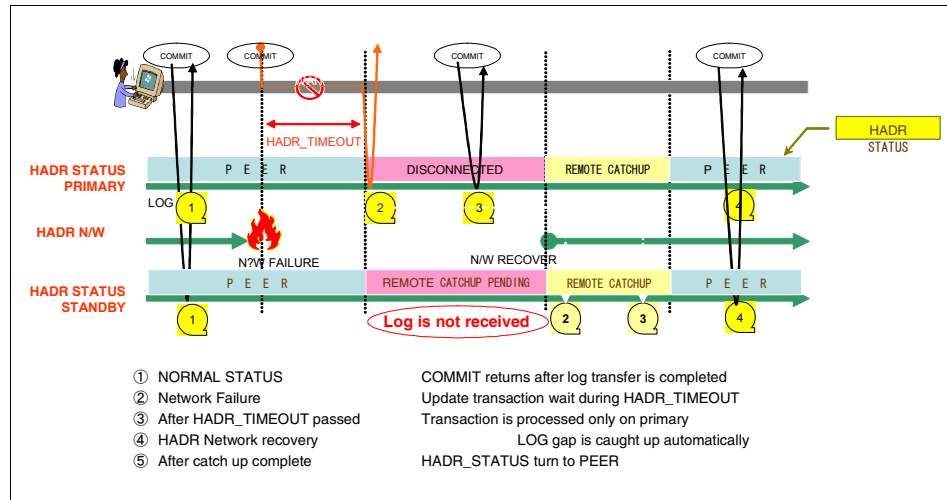


Figure 6-1 Behavior of HADR when the primary cannot transfer logs to the standby

## Network congestion

Network congestion can happen when the standby is behind in receiving and processing log data from the primary. This can cause the standby's log receive buffer to fill up, thereby preventing the buffer from receiving more log pages. This will cause a slowdown on the primary in all modes, including ASYNC mode.

In SYNC and NEARSYNC modes, if the primary database flushes its log buffer one more time, the data is likely to be buffered in the network pipeline, consisting of the primary machine, the network, and the standby database. Because the standby database does not have free log receive buffer (DB2\_HADR\_BUF\_SIZE) to receive it, it cannot acknowledge. So the primary will be blocked, waiting for the acknowledgement. Congestion is not likely to occur for SYNC and NEARSYNC modes as unless this log data is acknowledged, and no more logs will be sent by the blocked primary.

In ASYNC mode, the primary will continue to send logs until the network pipeline fills up. The network pipeline consists of the TCP buffers on the primary machine, the TCP buffers on the standby machine, the network path between the primary and the standby, and the HADR standby log receive buffer. When the primary cannot send any more logs, congestion can occur.

Large jobs like table reorganization can flood the standby and cause congestion as the standby database is replaying log records that take a long time to replay. If the primary load has sharp peaks, they might be absorbed by a larger standby buffer.

You can mitigate the congestion problem by tuning the TCP/IP network to increase TCP/IP network buffering and also increase DB2\_HADR\_BUF\_SIZE registry variable to increase the log receive buffer on the standby. But a larger buffer would not help if the primary has a sustained log rate higher than what the standby can handle. Snapshot™ or db2pd will report connection status as “congested”. Congestion is reported by the hadr\_connect\_status monitor element.

### 6.3.8 Avoiding data loss in an HADR with HA cluster software

An HADR with HA cluster software implementation automates the HADR takeover process. If a network error is detected, the primary database will enter the disconnected state and let transactions proceed. This behavior maximizes the availability of HADR database. Although the transactions can still be committed during the time the database is in a disconnected state, these transactions will not reach the standby and exist only on the primary database.

If the primary database goes down before the network is recovered, the automatic take over process starts. Since the primary database is down, the standby will take over the primary by force. The log gaps are not handled and the log records that were not transferred to the standby database, yet will be lost on the new primary database. Figure 6-2 shows the behavior of a HADR database when the primary database cannot transfer logs to the standby database.

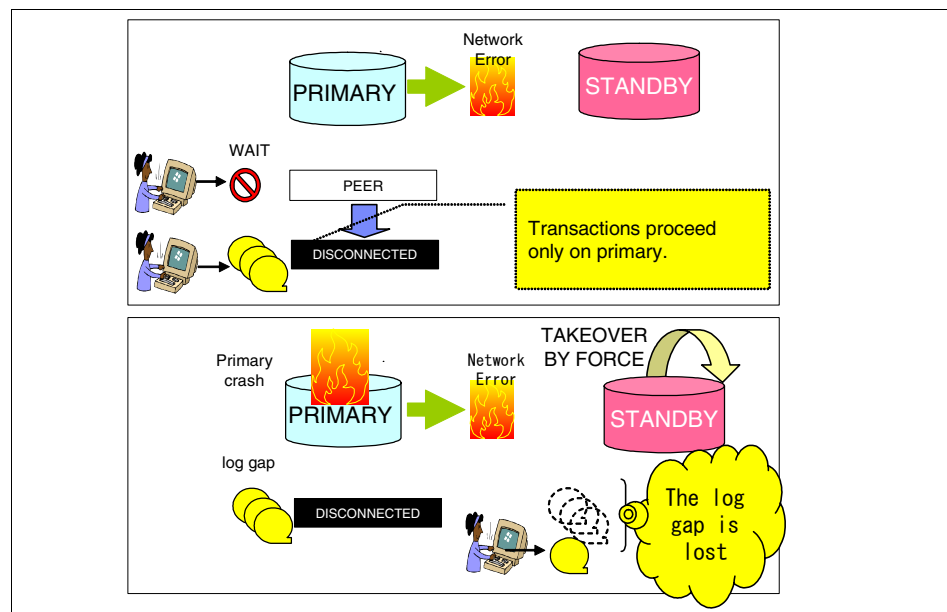


Figure 6-2 HADR behavior when the primary cannot transfer logs to the standby

**Note:** To reduce the possibility of network error, we recommend that you have duplicate networks for HADR communication.

To avoid data loss, the automated failover should only be performed if the HADR is in Peer state when the primary crashed. Before you execute HADR takeover, ensure that the HADR is in Peer state, which means that there is no log gap between databases in SYNC or NSYNC mode. ASYNC mode is not discussed here, since this mode does not guarantee the integrity of databases by its characteristics.

By monitoring the HADR status properly and adding a handling logic in an HACMP clustered environment, we can guarantee that the committed data will not be lost after the standby system took over the database. Figure 6-3 illustrates the concept of saving the committed data after the standby takeover.

The steps are as follows:

1. Monitor diag.log on the primary database.
2. Notify the HADR status.
3. Add integrity check logic to the takeover script.

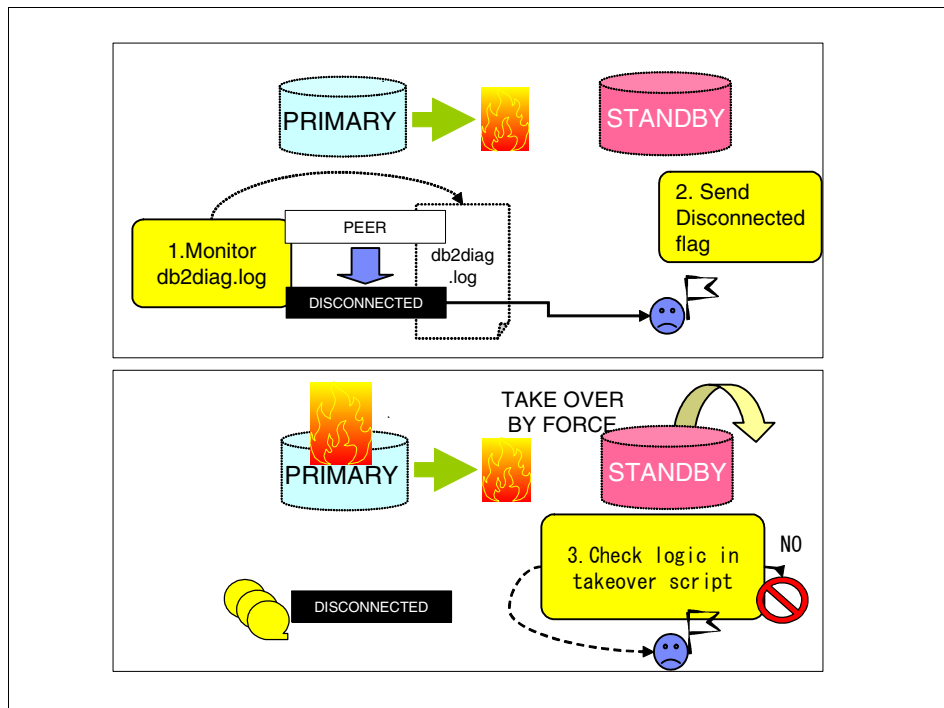


Figure 6-3 Avoiding data loss in an HADR with HA cluster software

1. Monitor diag.log on the primary database:

You have to detect the change of HADR status immediately when it turns into disconnected status. The best way to do this is by monitoring db2diag.log messages on the primary database. Example 6-1 shows a message in db2diag.log when state changed from peer to RemoteCatchupPending, which means that the primary database is disconnected from the HADR pair and can accept new transactions, leaving the standby database behind.

*Example 6-1 HADR status changed from Peer to RemoteCatchupPending*

---

```
2006-10-25-00.32.28.487034+540 E179519490C354    LEVEL: Event
PID      : 8056                TID   : 1        PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst            NODE   : 000        DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-RemoteCatchupPending (was P-Peer)
```

---

Example 6-2 shows a message in db2diag.log when status changed from RemoteCatchupPending to Peer.

*Example 6-2 HADR status changed from RemoteCatchupPending to Peer*

---

```
2006-10-25-00.35.47.823847+540 E179520902C353    LEVEL: Event
PID      : 8056                TID   : 1        PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst            NODE   : 000        DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-NearlyPeer (was P-RemoteCatchup)
```

---

```
2006-10-25-00.35.47.830199+540 E179521256C344    LEVEL: Event
PID      : 8056                TID   : 1        PROC  : db2hadrp
(SAMPLE) 0
INSTANCE: hadrinst            NODE   : 000        DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery,
hdrSetHdrState, probe:10000
CHANGE   : HADR state set to P-Peer (was P-NearlyPeer)
```

---

To keep tracking on db2diag.log, you can use the **tail** command. See Example 6-3 for a sample script (tail\_hadr\_status.ksh). When this monitoring script detects the changes of HADR status by tracking the messages in the db2diag.log, it writes the status to the file. For example, set “1” when HADR status is changed from peer to non-peer, and set “0” when HADR status is returned from non-peer to peer. This status flag file allows the standby system to know the HADR state when the primary is down.



#### Example 6-3 *tail\_hadr\_status.ksh*

---

```
tail -0 -f $DIAGLOG | awk '
/CHANGE \: HADR state set to P-RemoteCatchupPending \(was
P-Peer\) / {system("'"${WRITE_STATUS}"' 1")}
/CHANGE \: HADR state set to P-Peer \(was P-NearlyPeer\) /
{system("'"${WRITE_STATUS}"' 0')}
```

---

To bring this monitoring script up and running at any time, run the script with the *nohup* option on the primary node:

```
#nohup tail_hadr_status.ksh &
```

We provide a complete HADR monitoring script in A.3, “hadr\_monitor.ksh” on page 409.

#### 2. Notify the HADR status:

Since the takeover command is issued on the standby database, the standby system has to know the HADR status of the primary database to take proper actions. In other words, this status flag should be accessible from the standby node even though the primary node is crashed. In our example, the status flag is on the standby node and updated from the primary node using a remote command. Especially in an HACMP cluster, **c1\_nodectmd** is useful because this command can use any available network defined in HACMP cluster, which means **c1\_nodectmd** can be executed over one of the available network even if a network interface for HADR is down. This is because HACMP is configured with multiple network cards in many cases. Refer to Chapter 8, “DB2 and HACMP” on page 183 for more details.

Example 6-4 shows a code snippet of the update flag file script (*write\_status.ksh*). This script is included in the db2diag.log monitoring script *tail\_hadr\_status.ksh* as a function (*write\_status.ksh*). Parameter value *\$1* (0 or 1) is given by *tail\_hadr\_status.ksh*, and **c1\_nodectmd** writes it in the flag file on the standby node.

#### Example 6-4 *write\_status.ksh*

---

```
.....
```

```
usr/es/sbin/cluster/sbin/c1_nodectmd -cspoc '-n ${REMOTENODE}' "echo
$1 > $STATUS_FLAG"
```

```
.....
```

---

#### 3. Add integrity check logic to the takeover script:

In the takeover script, you need to add the logic to check the status flag before executing the TAKEOVER HADR command. It should failover only if the primary database had been in Peer state before it crashed. If the primary

database was not in Peer state before the moment it crashed, it will not execute takeover automatically and only notify that user intervention is required to bring the database online. The user could recover the primary server from the crash, or copy all logs from the primary node to the standby node, run local catchup, and then takeover on the standby database.

Figure 6-4 shows a summary of the events occurred and the actions taken to prevent data loss.

1. HADR Network failure.
2. HADR\_STATUS turns from peer to disconnected. The monitoring process catches the message in db2diag.log on the primary database and sends the flag file to the standby node.
3. Transactions are proceeded only on the primary database, not transferred to the standby.
4. The HADR network recovers from the error and log catch up is automatically run.
5. After completing catch up, HADR status returns to Peer state. The monitor script catches the message in db2diag.log on the primary database and sends the flag file to the standby node notifying that the HADR status had been back to peer status.
6. Meanwhile the flag is set to non-peer status “1” in event and kept the same status till event 5. We do not want to execute takeover by force because there is the possibility of data loss. We can add the logic in the takeover scripts to check this flag before issuing the TAKEOVER HADR command with the FORCE option.

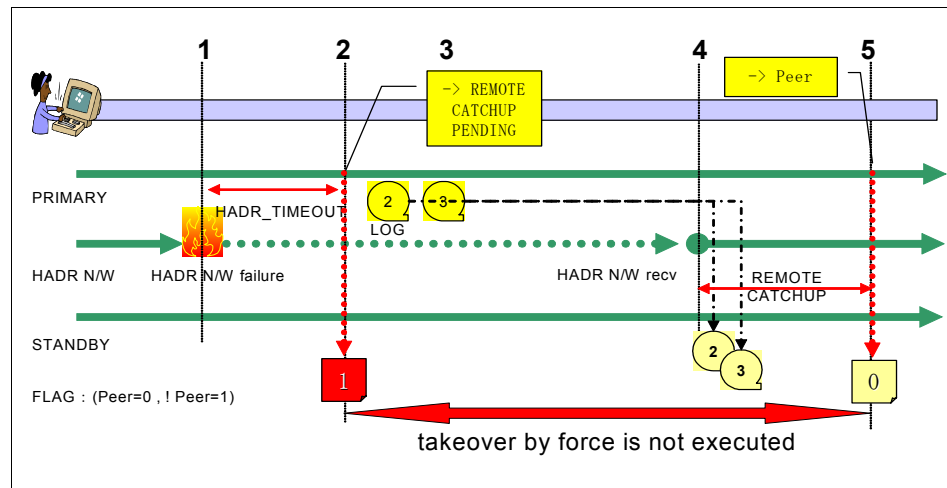


Figure 6-4 The behavior of the implementation

### 6.3.9 Index logging

By default, the index build creates a log “create object” record, but index pages within the object are not logged. The index is then marked as “pending rebuild” and will be rebuilt once recovery is complete. This is undesirable for HADR, as a standby will have pending build indexes that will be rebuilt when someone accesses them. On each takeover you will have indexes marked as bad, and will need an index rebuild. Index rebuild can take a long time after failover.

The LOGINDEXBUILD database configuration parameter must be set to ON to establish a rule at the database level to log all indexes. This controls whether to log index build or not. Index update is always logged. This parameter applies at index creation, recreation, and table reorganization. The default is OFF. We recommend setting this parameter to ON for HADR databases.

There is a table level log index attribute that overrides the database level configuration parameter. This table level parameter is set with the ALTER TABLE statement LOG INDEX BUILD option, which can be set to the following values:

- ▶ ON: Logs index build
- ▶ OFF: Does not log index build
- ▶ NULL: Defaults to what LOGINDEXBUILD is set in database configuration

There might be an impact to transaction performance if LOGINDEXBUILD is ON. The impact on performance depends on the amount logs generated and number of indexes rebuilt or reorged, the HADR synchronization mode, and the network availability and the standby's ability to keep up with the primary. When the LOGINDEXBUILD is ON, issuing a REORG command on large number of tables that also involves reorging indexes could impact transaction performance.

The database manager and database level configuration parameter INDEXREC specifies when the invalid index is to be re-built. Refer to “DB2 HADR configuration parameters” on page 138 for more details on the values for this parameter.

### 6.3.10 Read on the standby

HADR has certain restrictions; the standby database is not a full functional database. You cannot update the database from the standby, nor can you issue any read-only queries on the standby. A fully accessible standby database would provide businesses with an opportunity for data mining without impacting users' access on the primary database.

If there is a customer requirement to read from a standby, then the workaround is to use VERITAS Storage Foundation for DB2 HADR. For more details on this implementation, read the white paper, *Accessible Standby Database in IBM DB2 UDB HADR Environment Using VERITAS Volume Instant Snapshots*, at the Web site:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/vrts-hadr.pdf>

This article describes the design and implementation of a solution for providing a fully accessible point-in-time copy of the standby database in a DB2 UDB HADR environment. Although this copy of the standby database is fully accessible, it is separated from the HADR pair so updates made to it are not automatically applied to the main (HADR) pair of database copies.

This solution leverages a new feature called *instant space optimized volume snapshot* by which a single point-in-time copy of the storage can be created with only a small amount of additional disk space. Only storage for changed blocks is required. This article discusses how to activate a point-in-time copy of the standby database without impacting the active HADR databases. The space optimized volume snapshot feature enables the capability to produce a clean copy of the standby database without a WRITE SUSPEND on the primary database during the snapshot procedure.

Figure 6-5 illustrates the Veritas Storage foundation for DB2.

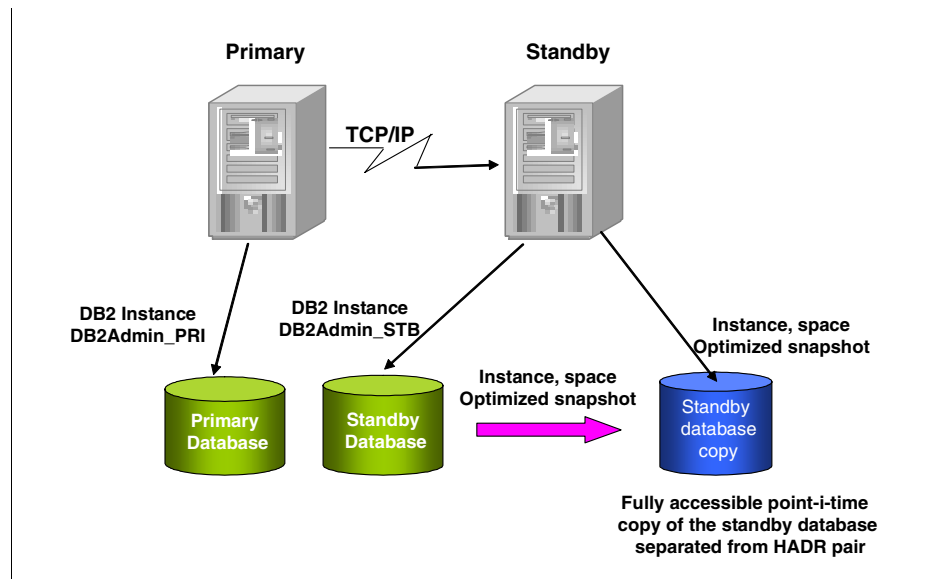


Figure 6-5 Veritas Storage foundation for DB2 overview

### 6.3.11 Backup from standby image with Flash Copy

With the Storage Copy function, you can utilize the ability to back up the HADR standby database without impacting the performance of the primary database.

Here we outline the steps to take a backup image from the standby database:

1. Database backup on the standby server:

Suppose you have another storage area for the snapshot of the standby database, and FlashCopy® is configured between the storage area for the standby database and the snapshot. Do the following steps:

- a. Deactivate the database on the standby database.
- b. Stop the database instance.
- c. Unmount all the file systems and varyoff volume groups. To ensure the consistency of the snapshot image, no write activity is allowed during FlashCopy logical copy. So it is preferable to unmount file systems (and varyoff volume groups).
- d. Use a Storage Copy function such as Flash Copy to take a snapshot of the storage image from the standby database, which includes the database directory and table space containers. Basically, the logical copy completes in a moment before waiting for the physical copy of whole storage area. With IBM Total Data Storage, once the logical copy is finished, you can activate standby image again.
- e. Activate (varyon) volume groups, mount file systems, and activate the database. Note that when the standby database is deactivated, you might not have the latest standby database.

2. Restoring to the primary server:

If you have to restore the database from this backup to the primary server, you must do the following steps:

- a. Restore all the DB2 related files to the correct original location on the primary server from this backup image. The restored database should be in database rollforward pending state already.
- b. Make all the log files available (including the active log files and archived log files from the primary server) to be processed.
- c. When the backup image was made on the standby server, its HADR status was on standby mode. Also, the HADR configuration parameters are copied from the standby database. They should be updated for the primary database.
- d. Apply the log files from the original primary database to the restored database.

- e. Restart the HADR (assuming that the standby database is still up and running).
3. Restoring to the standby server:

If the standby database becomes unusable due to a hardware problem, the split backup can be restored to the original standby database. No special configuration is required. Once the standby database files have been restored, start HADR as standby. The standby database should now go into catchup mode, with the primary database server retrieving the log files and shipping the log records over to the standby database until it reaches Peer state.

For more detail, see Chapter 7 of the document listed at the following Web site:

<ftp://submit.boulder.ibm.com/sales/ssi/sa/st/e/0rwhr-6h3ppe/HADR-in-an-SAP-implementation-04Aug2005.PDF>

### 6.3.12 Replicating load data

When LOAD utility is used to move data into a table, the data is not included in the log file. There are some special requirements to replicate load data in the HADR environment. The basic rule is that the load operation is replicated only if there is a copy of the loaded data available for the standby to obtain the data.

The copy device must be available when the standby replays the load operation. The standby might attempt to access the copy any time after the primary completes the load. If a load operation is executed on the primary database with the NONRECOVERABLE option, the command will execute on the primary database and the table on the standby database will be marked bad. The standby database will skip future log records that pertain to this table. You can choose to issue the LOAD command with the COPY YES and REPLACE options specified to bring the table back, or you can drop the table to recover the space.

For more details, see DB2 Information Center: **Administering → database systems → High Availability → Managing HADR → Database configurations for HADR**, and search for “Load operations and HADR”.

The following methods can be used for replicating load data to the standby database.

► Using NFS shared disk:

You can utilize network file system (NFS) to share the load copy image on both primary and standby nodes. See Figure 6-6. On the standby node, be sure to mount the NFS shared directory to the exactly same point as the primary can see.

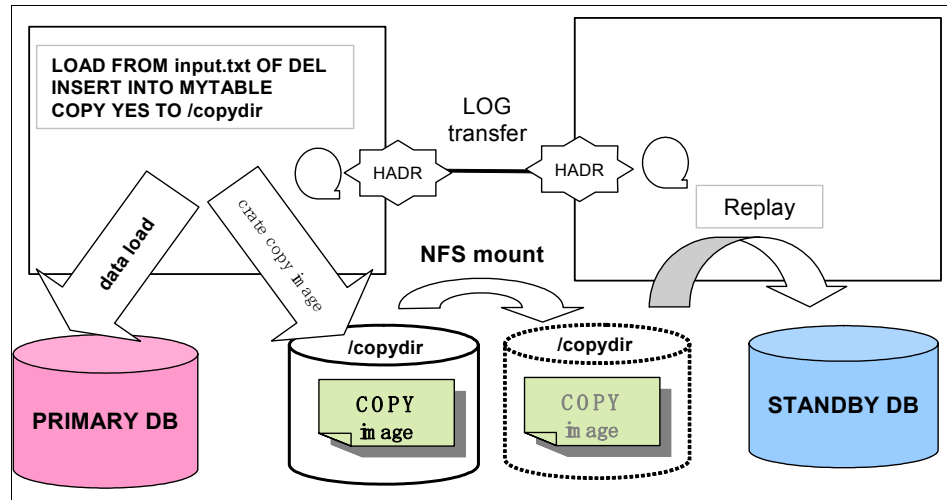


Figure 6-6 Using NFS to share data

2. Pause (Deactivate) the standby database during transferring load copy:

If you transfer the copy image by some means such as filecopy or physical tape media, we recommend that the standby be stopped (db2 deactivate db) before the primary runs the load. Once the primary finishes the load and copy is in place on the standby, restart the standby (db2 activate db). It will then reconnect and replay the load.

Make sure that instance owner has proper access permission to the copy file. For example, in AIX system, the permissions of copy image is set to 640 (rw-, r--, ---), instance owner of the standby database should have same UID (User ID) or belong to the same GID (Group ID) so that the copy image file generated by the primary instance is also accessible from the standby instance. If it is not accessible from the standby instance, the table space that includes the table with loaded data will be marked as restore pending and cannot accept any more updates. You'll have to rebuild the standby database from the primary.

### 6.3.13 Log archive and HADR

Log archive only happens on the primary. If the primary and the standby use independent log archive devices, with takeover back and forth, some log files will be archived on one device and others on the other device. Log is never archived on the standby database even though you set up USEREXIT, LOGARCHMETH1 or LOGARCHMETH2. The standby writes logs it receives from the primary into its local log files. To avoid filling up the log path, it recycles log files once a file is no longer needed for the purposes like crash recovery and takeover readiness.

For more information, see DB2 Information Center: **Administering → database systems → High Availability → high availability disaster recovery overview → Standby database states.**

### 6.3.14 Database restore consideration

Following are some considerations for database restore:

- ▶ Redirected restore is not supported.  
“Redirected” here means redirecting table space containers. Database directory (restore database ... to) and log directory (restore database ... newlogpath ...) changes are supported. Table space containers created by relative paths will be restored to paths relative to the new database directory.
- ▶ Restoring the standby database has certain requirements.  
When creating the standby database by restore from primary database, the restore command(s) should leave the database in rollforward pending state, otherwise, start HADR as standby will fail with HADR error SQL1767N start HADR cannot complete. Reason code = "1". Be sure *not* to specify the option, *without rollforward*, when you rebuild the standby database.

## 6.4 Restrictions

Here we summarize the HADR restrictions:

- ▶ HADR is not supported in a partitioned database environment.
- ▶ The primary and standby databases must have the same operating system type and the same version of the DB2 database system, except for a short time during a rolling upgrade. It is recommended to have the same operating system version, but is not mandatory. A different FixPak is allowed because it is needed for a rolling upgrade, but is not recommended for normal operation. When FixPaks are different, the primary cannot be a later version, because a later primary might generate log records the standby cannot understand.



- ▶ The DB2 database system release on the primary and standby databases must be the same bit size (32 or 64 bit).
- ▶ Database seed must be identical. The primary and the standby must originate from the same database.
- ▶ The same storage path is required on the primary and the standby. This is required so that table spaces can be replicated. The same storage path is often achieved by symbolic links.
- ▶ Reads on the standby database are not supported. Clients cannot connect to the standby database.
- ▶ Log archiving can only be performed by the current primary database.
- ▶ Self Tuning Memory Manager (STMM) can be run only on the current primary database.
- ▶ Backup operations are not supported on the standby database.
- ▶ Circular logging is not supported.
- ▶ Infinite logging is not supported.
- ▶ Non-logged operations, such as changes to database configuration parameters and to the recovery history file, are not replicated to the standby database. LOBs larger than 1 GB can only be defined as not logged. LOBS which are not logged are not replicated, but LOB space is allocated on the standby. The LOBs on the standby will have the right size, but the content will be binary zero.
- ▶ Use of Data Links is not supported.
- ▶ HADR does not support the use of raw I/O (direct disk access) for database log files. If HADR is started through the START HADR command, or the database is activated (restarted) with HADR configured, and the raw logs are detected, the associated command will fail.





## DB2 and system upgrades

In this chapter we focus on performing DB2 FixPak applies with the no-downtime benefit that HADR brings, and DB2 version upgrades or migrations with HADR databases. We also explore the considerations to be made when performing changes to other system components that might require DB2 or the whole server to be recycled. Lastly, we provide an overview of what to keep in mind when updating certain database configuration parameters.

We cover the following topics:

- ▶ DB2 FixPak rolling upgrades
- ▶ Version updates
- ▶ OS and application and database configuration parameter updates

## 7.1 DB2 FixPak rolling upgrades

HADR gives you high availability while applying DB2 FixPaks through rolling upgrades. Your database down time need only be literally momentary while you are switching roles between your database servers. With properly written applications using standard retry logic and handling of message SQL30108N if you are using Automatic Client Reroute (ACR), this effectively means no loss of data and no real downtime.

In this section we provide examples of a FixPak apply on a Windows server to illustrate the procedure through Graphical User Interface (GUI) tools. We also give you an example of a rolling upgrade performed with the DB2 Command Line Processor (CLP).

These are the steps to perform a rolling upgrade:

1. Check your system's HADR status.
2. Apply FixPak on the standby system.
3. Switch roles from the standby system.
4. Apply the FixPak in the old primary system.
5. Switch roles back to the original primary.

### 7.1.1 Applying DB2 FixPak on Windows operating systems

In this section, we describe the procedure to apply DB2 FixPak on Windows. Figure 7-1 illustrates the rolling upgrade performed in our Lab environment.

We used two servers, PUGET and FAROE. Both systems have the database SAMPLE, where PUGET is initially set up to be the primary HADR server, and FAROE is initially the standby server. In this example we upgraded from DB2 for LUW Version 8.2 FixPak 12 to FixPak 13, using the Graphical User Interface (GUI) as much as possible.

Figure 7-1 illustrates the rolling upgrade we performed.

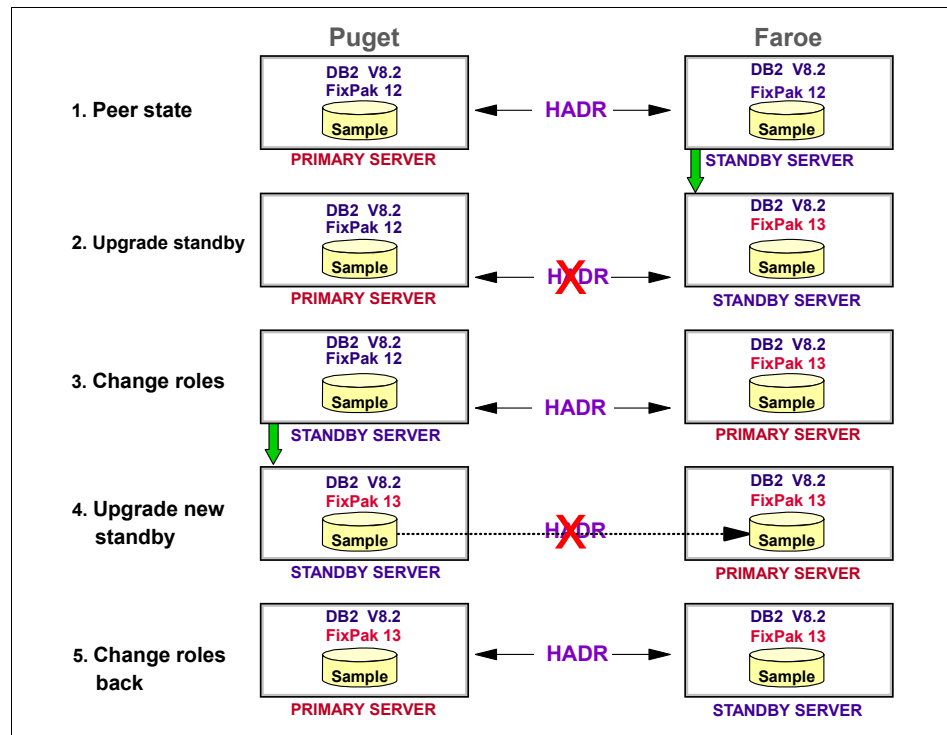


Figure 7-1 Rolling upgrade

The steps to apply the FixPak are as follows:

#### 1. Check your system's HADR status.

This first step is a pre-requisite check; first your DB2 administrator ID (db2admin) must be a member of the local Administrators group. Second, we recommend that the HADR status be in Peer state for both servers. Check HADR status with db2pd, get snapshot, or the Manage HADR window. See Chapter 4, "HADR administration and monitoring" on page 93 for the details of using these utilities.

To open a Manage HADR window, from the Control Center, expand the object tree down to the database object, and right-click on the database name. Select **High Availability Disaster Recovery** → **Manage ...**

Example of the GET SNAPSHOT command:

**db2 get snapshot for database on sample**

Look for the following lines in the output:

```
HADR Status
Role           = Standby
State          = Peer
```

2. Apply the FixPak code on the standby server (FAROE in our test case). Refer to the Readme.txt file located in your FixPak directory for more details on the actual FixPak apply. The basic FixPak installation steps are as follows:

- a. Deactivate your database with the DEACTIVATE DATABASE command from the CLP.

**db2 deactivate database** *sample*

Then proceed to stop your instance from the CLP or DB2 Control Center. From the Control Center, expand the object tree down to the database object. Right-click the database name, and then click **stop**. This will stop the instance as well as the database.

- b. Stop all DB2 prefixed services in the Microsoft Management Console Services Snap-in by:

**Start** → **Run**, enter **services.msc**, and click **Enter**

- c. Change to the folder where the unzipped files are located. The setup command is located under the folder with the product abbreviation. For example, DB2 Enterprise Server Edition would be under ESE.

To start the DB2 Setup wizard, double click the setup.exe file. The DB2 Setup wizard GUI Launchpad opens.

- d. You might receive warning boxes for services still running, go ahead and click **Yes** for the wizard to stop them.
- e. Setup is complete. Click **Finish**.
- f. A pop-up appears asking to restart Windows. Click **Yes**. After the reboot, First Steps window appears, go ahead and close it by clicking **Exit** in the left hand menu pane.
- g. From the Microsoft Management Console Services Snap-in confirm DB2 services are running by:

**Start** → **Run**, → enter **services.msc**, click **Enter**. The following two processes should be started.

- DB2-DB2-0 (or whichever instance you are using for HADR)
- DB2DAS - DB2DAS00

**Note:** At this time you will not be able to issue the bind commands required in the FixPak apply, because the database is in rollforward pending state. The binds will be addressed later in the procedure, in step 4.b.

h. Activate the database with the CLP on the standby system. For example:

**db2 activate database sample**

3. Switch roles from the standby system.

In our example, we switch roles and the FAROE server becomes the new primary and PUGET is now the new standby.

On the new standby system, open the DB2 Control Center, expand the object tree down to the database object, right-click the database name. Select **High Availability Disaster Recovery → Manage ...**

a. In the Manage High Availability Disaster Recovery (HADR) window, click **Takeover HADR ...** as shown in Figure 7-2.

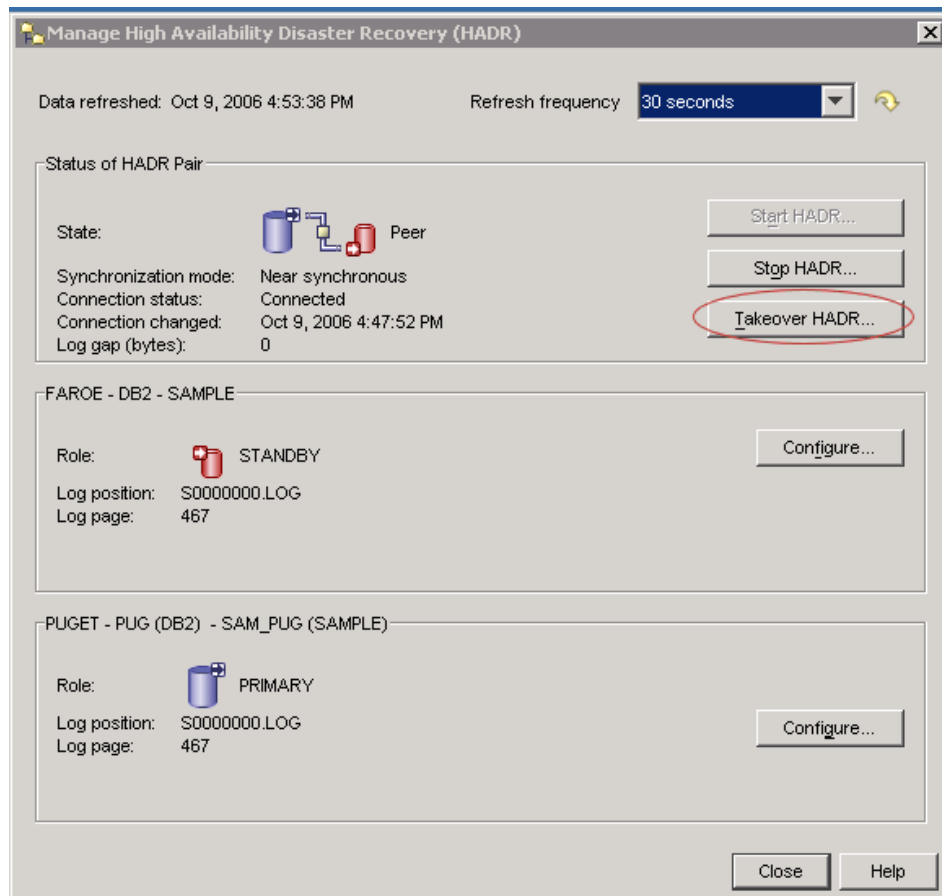


Figure 7-2 HADR takeover

- b. To issue role switch, select **Switch roles** in the Takeover HADR window, click **OK**. See Figure 7-3.

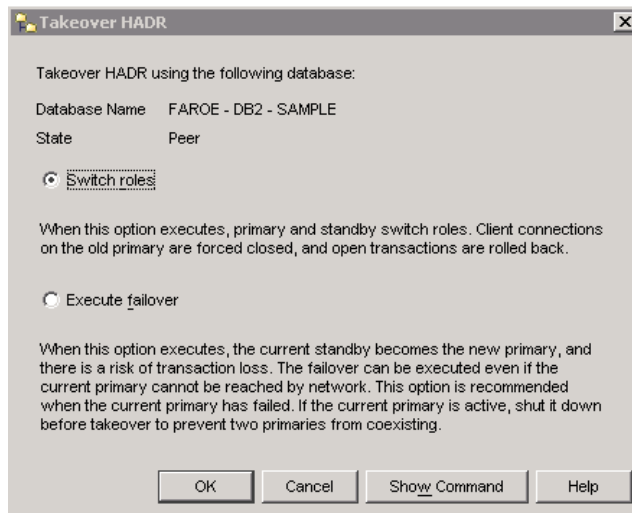


Figure 7-3 Switch roles

- c. A DB2 message should come up to confirm success. See Figure 7-4. Click **Close**.

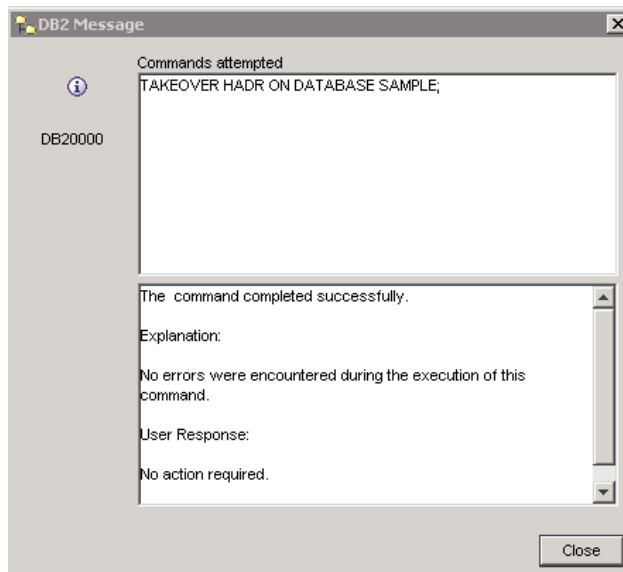


Figure 7-4 Takeover successful message



- d. Manage High Availability Disaster Recovery (HADR) window should now show you that the standby machine now is running as primary on this server. In our example, FAROE is now the primary, see Figure 7-5.

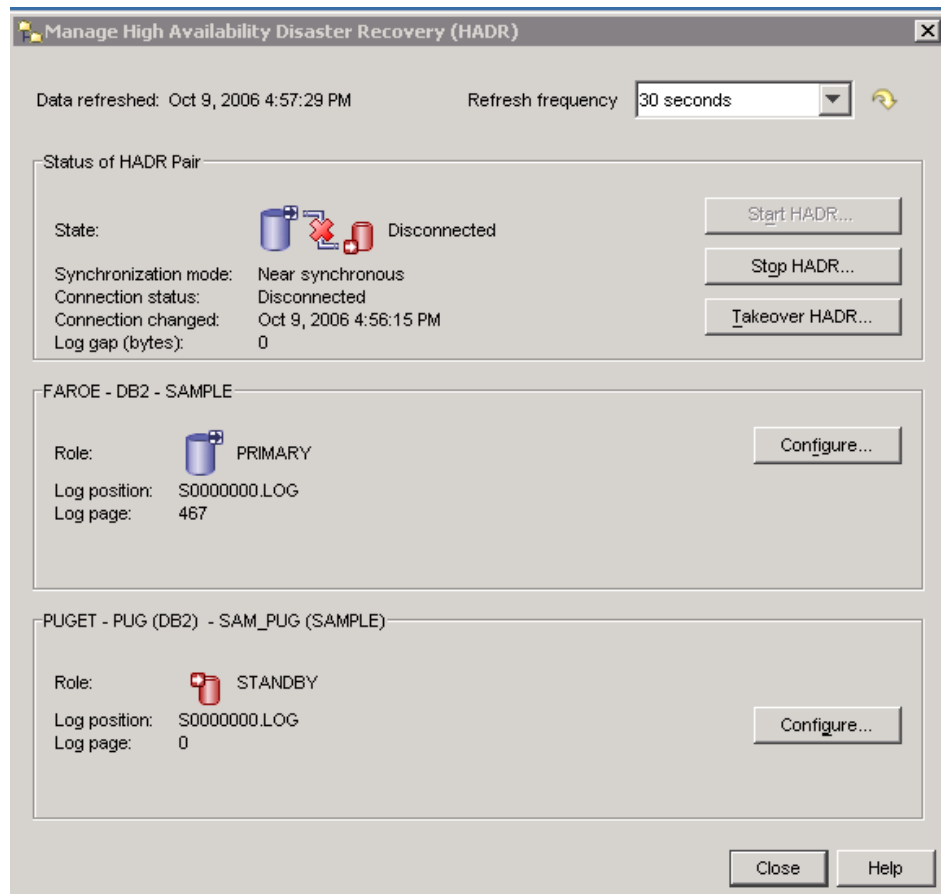


Figure 7-5 Successful takeover

Note that HADR supports the standby running newer level than the primary, but not the other way around. Therefore, once the takeover is complete, the new standby (old primary) database will be deactivated, since the new primary has higher later DB2 level.

4. Apply the FixPak in old primary system, PUGET in our example.
  - a. Repeat step number 2 on the new standby server. In our example, we perform the FixPak apply on PUGET.

- b. The binds cannot be performed from a standby database so you have to connect to the current primary database — FAROE for our example. Once connected, execute the binds as listed in the FixPak readme downloadable text or HTML file (found in the same place where you download the actual FixPak code). The binds are logged, so HADR replays them on the standby to complete the FixPak apply.
5. (Optional) Switch roles back to the original primary if you want to restore your original server roles.
  - a. Your HADR status must be Peer to ensure all logs have been applied.
  - b. Repeat step number 3 on the original primary server, currently new standby server, PUGET in our test case.

## 7.1.2 Applying DB2 FixPak using command line

This section gives an example of the rolling fix pack upgrade in a command line environment. The test case we used involved two Linux servers LEAD (set up as the primary), and POLONIUM (set up as the standby). We performed a rolling upgrade from 8.2 FixPak 12 to 8.2 FixPak 13.

1. Check your system's HADR status.

We highly recommend that your HADR systems are already in Peer state. You can check your HADR status by db2pd, get snapshot, or Manage HADR window if you have X11 working.

Example of the GET SNAPSHOT command:

**db2 get snapshot for database on *sample***

Look for the following lines in the output:

```
HADR Status
Role           = Standby
State          = Peer
```

2. Apply FixPak on the standby system.
  - a. Deactivate the database on the standby server from the CLP. The standby is POLONIUM for our example:
 

**db2 deactivate database *sample***
  - b. Stop DB2 from the CLP in preparation for the FixPak apply:
 

**db2stop**
  - c. Apply FixPak code. Refer to the readme.file for the FixPak you are installing.
  - d. Start DB2 from the CLP on the standby, POLONIUM in our example:
 

**db2start**

- e. Activate the database on the standby from the CLP.

**db2 activate database** *sample*

**Note:** At this time you will not be able to issue the bind commands required in the FixPak apply, because the database will still be in rollforward pending. The binds will be addressed in step 4b.

3. Switch roles from the standby system.
  - a. From the CLP on the standby database, issue the **takeover hadr** command. In our test case, from POLONIUM:  
**db2 takeover hadr on database** *sample*  
After the take over, your HADR Connection status will be *disconnected* because of the different levels of DB2.  
Note that HADR supports the standby running newer level than the primary, but not the other way around. Therefore, once the takeover is complete, the new standby (old primary) database will be deactivated since the new primary has higher later DB2 level.
4. Apply the FixPak in the old primary system.
  - a. Repeat step number 2 on the new standby server. In our example, we perform the FixPak apply on LEAD.
  - b. The binds cannot be done from a standby database, so you need to connect to the current primary database — POLONIUM for our example. Once connected, execute the binds as part of the DB2 FixPak apply. The binds are logged, so HADR replays them on the standby to complete the FixPak apply. Refer to the FixPak Readme file for bind instructions.
5. (Optional) Switch roles back to the original primary.
  - a. Your HADR status must be Peer to ensure that all logs have been applied.
  - b. Repeat step 3 on the original primary server, currently new standby server, LEAD in our test case.

## 7.2 Version upgrade

A rolling upgrade is not supported between DB2 version or major subversion upgrades, such as version 8.2 to 9.1. For these events, plan for a database outage while the primary database is updated. In this section we give an example of a DB2 migration on the Windows operating system and a command line.

In order to minimize down time on your HADR servers, we recommend the version upgrade process as illustrated in Figure 7-6.

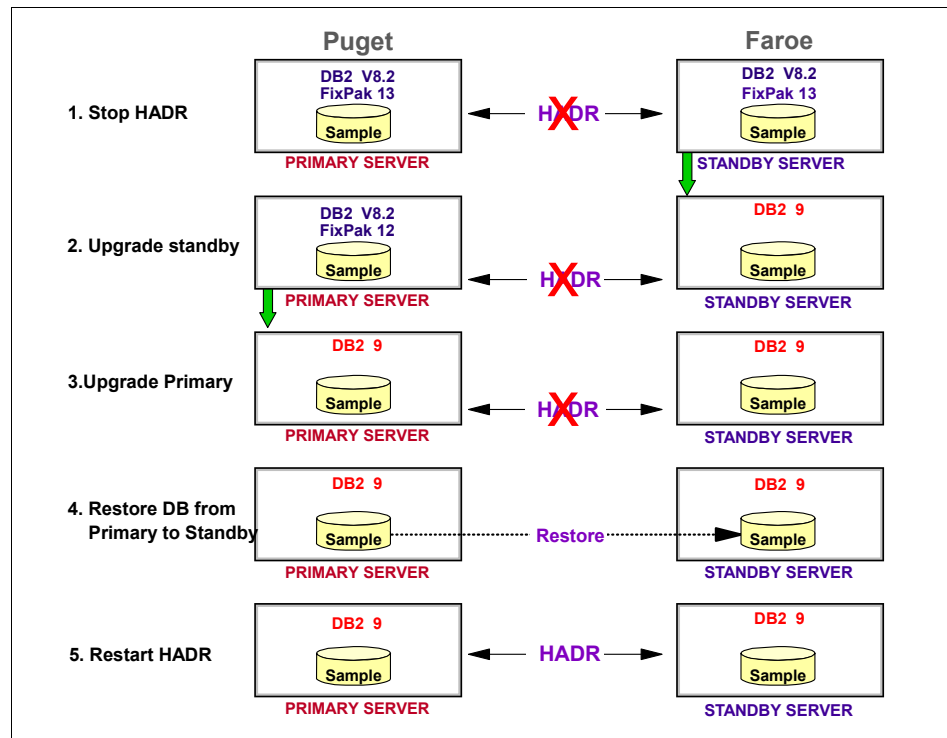


Figure 7-6 Version upgrade

## 7.2.1 Windows version upgrade

Next we describe a step-by-step upgrade from DB2 version 8.2 to version 9.1 on a Windows server.

Follow this procedure:

1. Upgrade the standby server.
  - a. Make sure that your HADR is a Peer status by db2pd, snapshot, or HADR manage window. See Chapter 4, “HADR administration and monitoring” on page 93 for details on command usages.
  - b. Run the DEACTIVATE database command on the CLP. For example:  
**db2 deactivate database sample**

Then proceed to stop your instance from the CLP or DB2 Control Center. From the Control Center, expand the object tree down to the database object, right-click the database name, then click **stop**. This will stop the instance as well as the database.

- c. Stop all DB2 prefixed services in the Microsoft Management Console Services Snap-in by: **Start** → **Run**, enter **services.msc**, then click **Enter**.
- d. Extract the DB2 v9 install zip to your preferred directory.
- e. Run Setup.exe.
- f. The first pop-up is Install a Product (option to install, or to migrate — we chose migrate). See Figure 7-7.

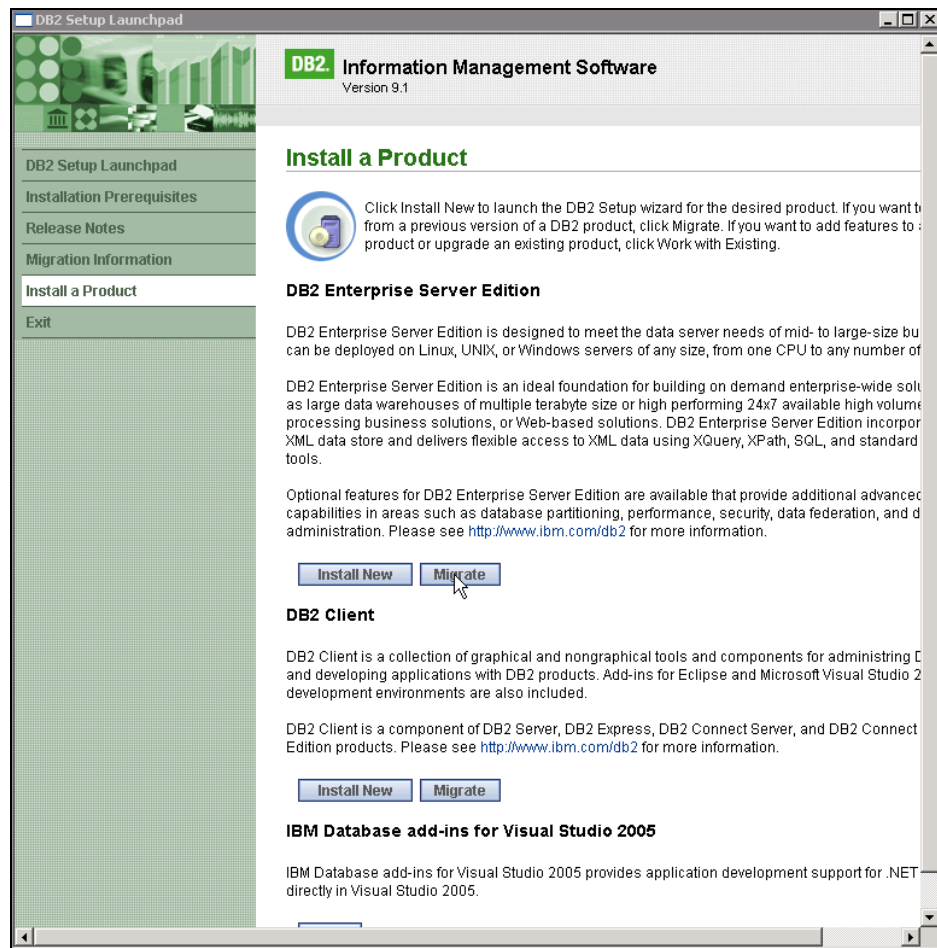


Figure 7-7 DB2 Setup Launchpad

- g. For the Warning pop-up (saying that your system has a previous level of DB2 installed), just click **Yes** to continue (Figure 7-8.)

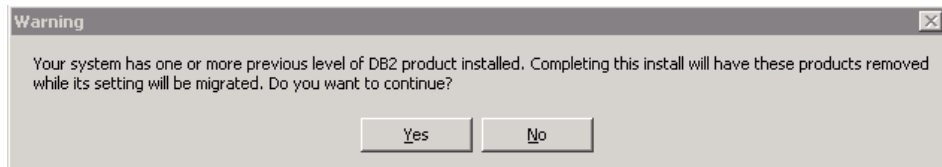


Figure 7-8 Warning previous version

- h. For the next Warning pop-up (saying that your system may have local databases - ignore/irrelevant on standby), just click **Yes** to continue (Figure 7-9).

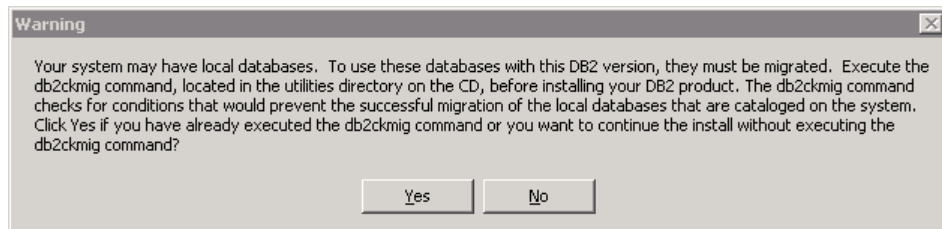


Figure 7-9 Warning for db2ckmig run

- i. The main window pop-up will go through the actual installer. Click **Next**.
- j. Click the radio button to accept the licence. Click **Next**.
- k. Choose your install type. Click **Next**.
- l. Save settings in a response file (optional) and install. Click **Next**.
- m. Select destination folder. Click **Next**.
- n. DB2 copy name - level of abstraction for the destination folder for a given version. Click **Next**.
- o. Set user information for default instance - specify the domain account.

**Note:** If you do not want Windows to try to create an ID on the domain, remove the ID in the domain field.

- p. Enable operating system security (optional — this alters the filesystem access control list so that only users in the DB2 groups can access the DB2 files/folders) and click **Next**.
- q. Start copying files and create a response file (summary of steps). Click **Finish**.

- r. Setup is complete. There is a reminder to do post install steps. Click **Install**.
  - s. The First Steps window appears. Click **Exit**.
2. Upgrade the primary server:
- Move over to the primary server PUGET in our test case. Because of the different version levels of your two servers, you will not be able to switch roles to the standby, so your database down time starts here.
- a. From the CLP, issue the DEACTIVATE command:
 

```
db2 deactivate database sample
```

The deactivate database command brings your database into a consistent state in preparation for the migration check.
  - b. Issue the db2ckmig command:
 

```
db2ckmig sample -l c:\dir\db2ckmig.log
```
  - c. Run a backup of your database, either through the Control Center Backup wizard or CLP. To issue the backup from the Control Center, expand the object tree down to the database object, right-click the database name, then click → **Backup**. This backup is done in case a back out is needed from the version upgrade:
 

```
db2 backup database sample to c:\db2backup
```
  - d. Run the **db2support** command to get instance config information and other information for fallback if necessary.
 

```
db2support c:\work -cl 0
```
  - e. Refer to step 1.b through to 1.s to install the new code.
  - f. Issue the **migrate database** command to all databases in your instance. Note that this also applies to any tools database (used by the DB2 Task Scheduler) for example:
 

```
db2 migrate database sample
```
  - g. Run a backup of your database, which will be moved over to the standby server in order to re-establish the HADR. At this point we recommend that you take an online backup to minimize your outage, but an offline backup is viable as well. As in step 2.c you have the option to do the backup from the CLP or Control Center Backup wizard.
 

```
db2 backup database sample online to c:\directory
```
  - h. Move the backup image over to the standby to restore below in step 3.a, or place it in a location visible to the standby server.

### 3. Re-initialize the standby server.

Return to your standby (FAROE server in our test case), then do these steps:

- a. Restore the backup taken from the primary database PUGET server in our test case. You can either use the Control Center Restore wizard or the **restore** command on the CLP. To access the restore wizard from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Restore**.

**db2 restore database sample from C:\directory taken at  
yyyymmddhhmmss**

- b. Start HADR on the standby server (FAROE in our test case). To start from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Manage ...**. See Figure 7-10. Then click **Start HADR ...** and make sure to uncheck the primary and keep the standby checked. Click **OK**.

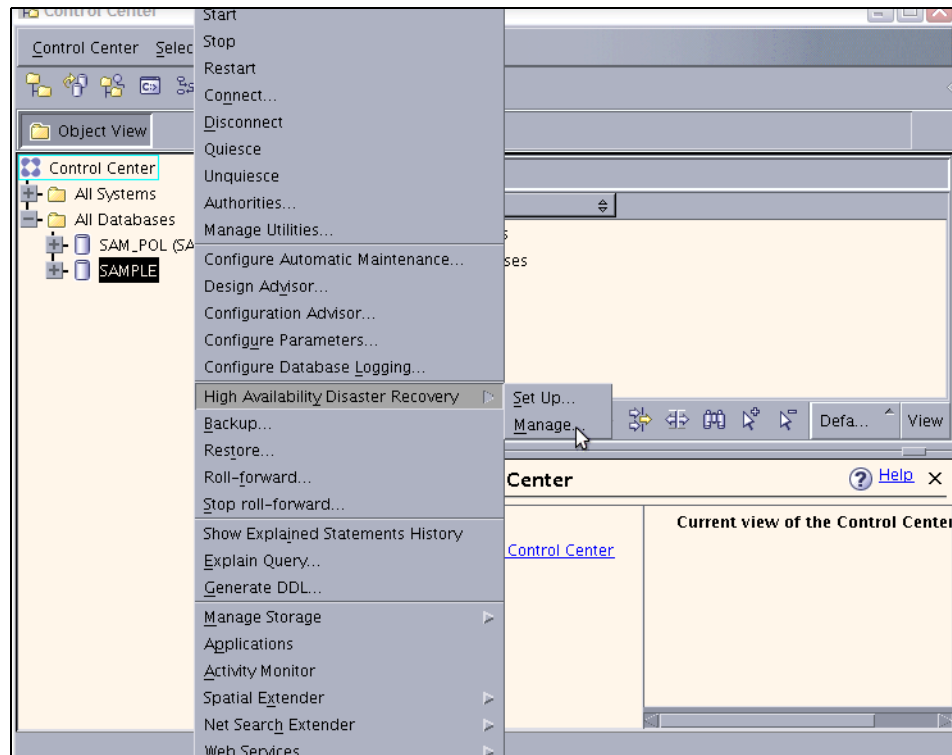


Figure 7-10 Manage HADR



- c. Now return to the primary server (PUGET in our test case) and start HADR on the database as the primary. To start from the Control Center, expand the object tree down to the database object, right-click the database name, then click **Manage ...**. See Figure 7-10. Click **Start HADR** and make sure to uncheck the standby and keep the primary checked. Click **OK**.

## 7.2.2 Version upgrade using command line

This section covers an example of a version upgrade on the command line environment. In our test case we migrated from version 8.2 to version 9.1 on a Linux environment. The servers used in our example are LEAD and POLONIUM both using the database SAMPLE.

### Procedure

The following steps are for upgrading DB2 version using commands:

1. Start the procedure on the standby POLONIUM.
  - a. Make sure that your HADR is a Peer status with db2pd, snapshot, or HADR manage window.
  - b. Deactivate your database:  
**db2 deactivate database *sample***
  - c. Stop HADR on the standby:  
**db2 stop hadr on database *sample***
  - d. In this step you have a choice to either drop the database on the standby POLONIUM or you can rollforward the database to bring it into a consistent state to do db2imigr. If you decide to drop the standby database, then you will need to reset the HADR database configuration parameters when you restore the backup from the primary.  
**db2 rollforward database *sample* to end of logs and complete**
  - e. Stop the database:  
**db2stop**
  - f. Stop the DB2 Admin Server from the DAS user id:  
**db2admin stop**
  - g. Perform DB2 code install. Follow directions from the *Quick Beginnings for DB2 Servers*, GC10-4246, and *Command Reference*, SC10-4226 for a **db2\_install** command line installation, or for a GUI interface with **db2setup**.

2. These are the steps to execute on the Primary database. At this point you will have an outage on the database.
  - a. Stop HADR on the primary - LEAD in our test case:  
**db2 stop hadr on database *sample***
  - b. Deactivate the database:  
**db2 deactivate database *sample***
  - c. Stop the DB2 Instance:  
**db2stop**
  - d. Stop the DB2 Admin Server from the DAS user id:  
**db2admin stop**
  - e. Perform code install - see step 1.g:
  - f. Start the DB2 Instance:  
**db2start**
  - g. Now you have to take a backup to move over to the standby in order to re-establish HADR. At this point we recommend that you take an online backup to minimize your outage, but an offline backup is viable as well. Here is an example command line syntax:  
**db2 backup database *dbname* online to /usr/db2/backup**
  - h. Move the backup image over to the standby database (POLONIUM in our test case). We use the example Secure Shell Copy (scp):  

```
scp SAMPLE.0.db2inst1.NODE0000.CATN0000.yyyymmddhhmmss.001
db2inst1@polonium:/usr/db2/backup
```
3. These are the steps to issue on the secondary database server - POLONIUM in our test case:
  - a. Restore your database on the standby server from the backup you transferred over in step 2.h:  
**db2 restore database *dbname* from /usr/db2/backup taken at yyyymmddhhmmss**
  - b. Start HADR on the standby (POLONIUM):  
**db2 start hadr on database *dbname* as standby**
4. These are the final steps on the primary (LEAD):
  - a. Start HADR:  
**db2 start hadr on database *dbname* as primary**

- b. Check HADR state until the connection succeeds and is approaching or at Peer state. See Chapter 4, “HADR administration and monitoring” on page 93 for the details of monitoring HADR.

## 7.3 Rolling OS/Application/DB2 configuration parameters

This section covers high level instructions for database and database manager configuration parameters, as well as any operating system (OS) and application upgrades. In order to keep your HADR systems optimal, you should apply changes to both systems as quickly as possible.

Note that when changing HADR database configurations, DB2 will not allow you to switch roles, so this will require a short database outage while you update the parameters on the primary server.

For the primary, this process is not required for those databases or database manager configuration parameters which are dynamic - that is, which require no recycling of the DB2 database or instance to take immediate effect. For the Istanbul, since you cannot connect to the database to perform dynamic configuration parameter updating, recycling the database is required to change configuration parameters. The listing of these parameters and which are dynamic can be found in *DB2 9.1 Performance Guide*, SC10-4222.

### Procedure

In our procedure we use the example environment of Server A and B. The DB2 database on Server A is initially the HADR primary and B is the standby.

1. On Server B (in standby mode):
  - a. Make sure that your HADR is in a Peer state with db2pd, snapshot, or HADR manage window.
  - b. Deactivate the database using Deactivate database command.
  - c. If necessary, stop the DB2 Instance. This applies to Operating System or non-DB2 changes which require a server recycle, in addition to Database Manager Configuration Parameters which require a DB2 Instance recycle.
  - d. Make the necessary changes to the hardware, software or DB2 configuration parameters.
  - e. Start the DB2 Instance if it was stopped.
  - f. Activate the database.

- g. Check HADR to ensure Peer state by db2pd, snapshot, or HADR manage window.
  - h. Switch roles of the primary and standby by issuing the **takeover** command.
2. On Server A (in standby mode):
- a. Direct clients to the HADR primary database on server B. This can be done by Automatic Client Reroute ACR.
  - b. Repeat step 1, optionally with step 1.h.



## DB2 and HACMP

In this chapter we explain how to integrate DB2 in a High Availability Cluster Multi-Processing (HACMP) environment. We provide the basic management concepts with recommendations and considerations to reduce the time consumed for failover.

We cover the following topics:

- ▶ Overview
- ▶ How DB2 works with HACMP
- ▶ Planning an HACMP cluster
- ▶ Setting up an HACMP cluster
- ▶ Considerations for the db2nodes.cfg file
- ▶ Tuning tips for quick failover

## 8.1 Overview

HACMP for AIX provides a highly available computing environment. HACMP facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. The primary reason to create HACMP clusters is to provide a highly available environment for mission-critical applications. In an HACMP cluster, to ensure the availability of these applications, the applications are placed under HACMP control. HACMP ensures that the applications remain available to client processes even if a component in a cluster fails. To ensure availability in case of a component failure, the HACMP software moves the application along with resources to another node in the cluster.

Here we list some common HACMP terms:

- ▶ **Topology:**  
The layout of physical components and connections defined in HACMP.
- ▶ **Cluster:**  
The group of nodes that work together closely for the purpose of enhancing availability of services.
- ▶ **Node:**  
Each server within the cluster definition. A *Service* (or *Primary*) node is designated as active to provide service to applications. A *Standby* node sits ready to take over if the service node fails.
- ▶ **Resource:**  
A *resource* is an object that is protected and controlled by HACMP. It may include the IP address to which clients access, file systems, or raw devices on shared volume groups, and the start and stop scripts to control applications.
- ▶ **Resource group:**  
This is a group of all the resources that need to be failed over from one server to the other. These include, but are not limited to, the following items:
  - The shared disks as defined in the volume groups. Raw devices or file systems are defined on them.
  - The IP address (*Service Address*) which the clients connect to.
  - The applications to be started to provide services. DB2 instance start and stop scripts are included in this definition.
- ▶ **Service address:**  
The IP address that provides services to clients. The service IP address is included in the resource group.

► Shared disk:

Shared disk is storage devices and volume groups connected to multiple nodes. Raw devices and file systems required to provide services are placed on them. From the perspective of DB2 in a non HADR implementation of HACMP, this would include the database directory, table space containers, database logs, and so on.

► Application server:

The application server is simply a set of scripts used to start and stop the applications running on HACMP node. One is an *application server start script* that starts up the applications which are a prerequisite to a provision of client services. The other is an *application server stop script* that stops applications before releasing resource groups. From the perspective of DB2, this would include scripts to start up and stop the database instance.

While there is only one main script to provide stop and start functionality for all applications and services, this work can easily be made modular by splitting the tasks into many subscripts which are called from the main script. For instance, with appropriate logic in the main calling script, subscripts can be called in a specific order according to file name, and can be dynamically added or removed from a subdirectory as required without changing the main calling script.

For more information about HACMP terminology, refer to *AIX HACMP v5.4 Master Glossary*, SC23-4867-08.

## 8.2 How DB2 works with HACMP

In this section, we focus on the HACMP cluster with databases created on shared disks. High availability is enabled by moving these disk resources (changing which node has control over them) and restarting database instance on a standby node when the service node has an outage. We call this classic type of cluster a *shared disk cluster*. Here we explain how DB2 works in a shared disk cluster controlled by HACMP with an example of a simple single partitioned instance.

Figure 8-1 illustrates HACMP topology and related DB2 database components. HACMP manages the group of resources required to provide a service, including shared disks where database components reside, the IP address for client service, and the application server handling the starting and stopping of the applications (including DB2 database instance).

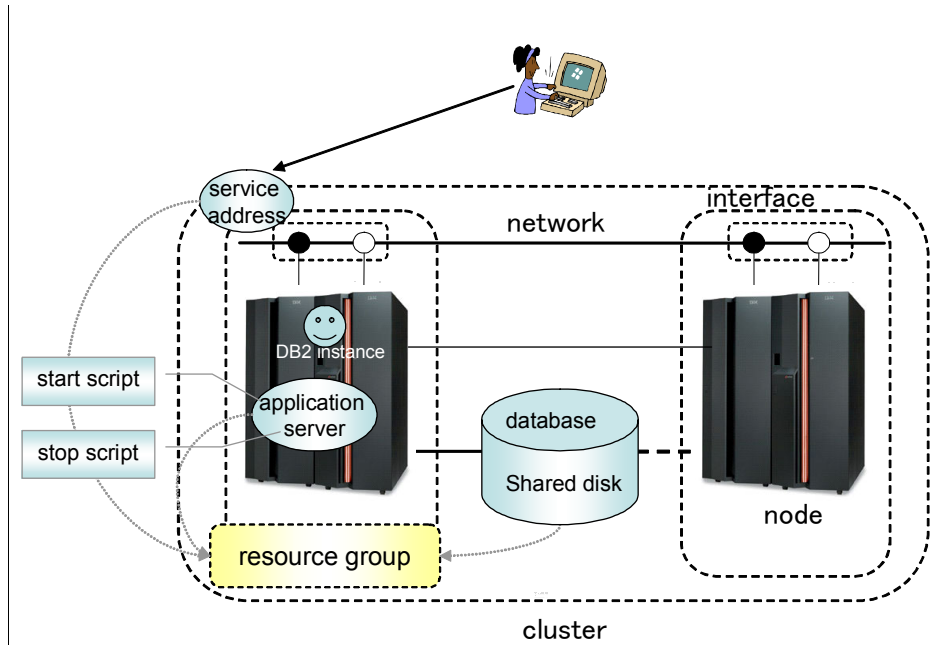


Figure 8-1 HACMP topology and related DB2 database components

HACMP continuously sends heartbeats between the nodes in the cluster to identify if and when the other one is down. HACMP handles resource group takeover from one system to the other as needed after a failure or fallback once the failed node is repaired. In a DB2 single-partitioned environment, the cluster takes the following actions:

► *Failover* (unplanned takeover):

If the service node fails, the HACMP standby node detects the service node outage. When the *keepalive* packets from all network routes are not received from the other node, the standby node starts to take over the resource group. This means that the standby node acquires the shared storage devices and the volume group where database components are created, and the service IP address through which clients communicate with the server. Once the resources are taken over, the start script defined in the application server is issued, which starts the DB2 instance along with other critical services and applications. Figure 8-2 shows how the resource group has been moved to a standby node during failover.



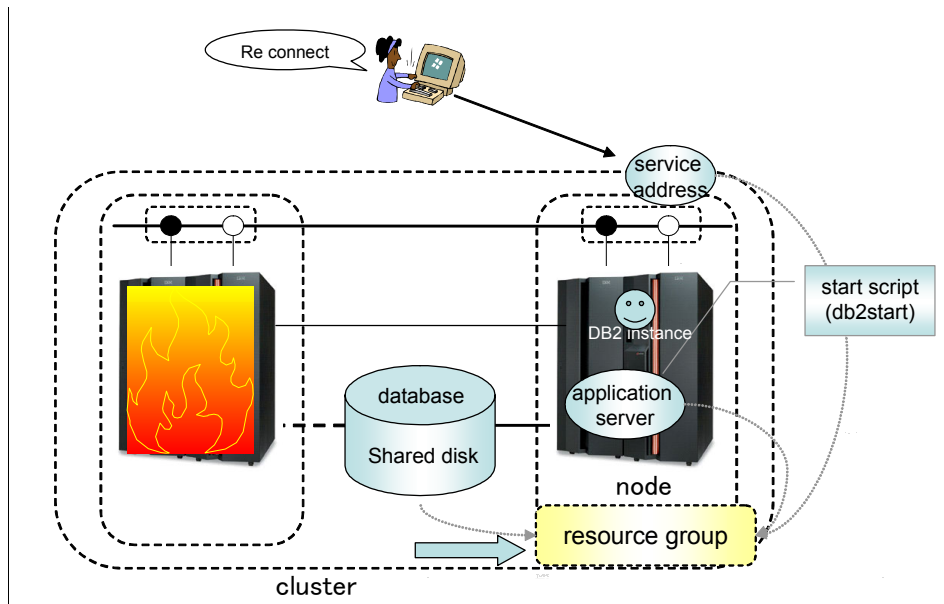


Figure 8-2 Failover (unplanned takeover)

► *Switch over* (planned takeover for maintenance):

You can switch over the resource group intentionally to the standby node for the purpose of machine maintenance. This is done by performing two HACMP management operations:

- Stop HACMP in takeover mode on service node.

The application server Stop script on the node that needs to release the resource group is invoked. The Stop script gracefully stops all the applications and processes which access the shared disk.

- Move the resource group from the service node to the standby node.

The HACMP process is still running on both nodes. HACMP releases all resources including shared disks and any service IP address related to the resource group. After released by the service node, the resource group is acquired on the standby node's side.

Figure 8-3 illustrates how the resource group moves during switch over.

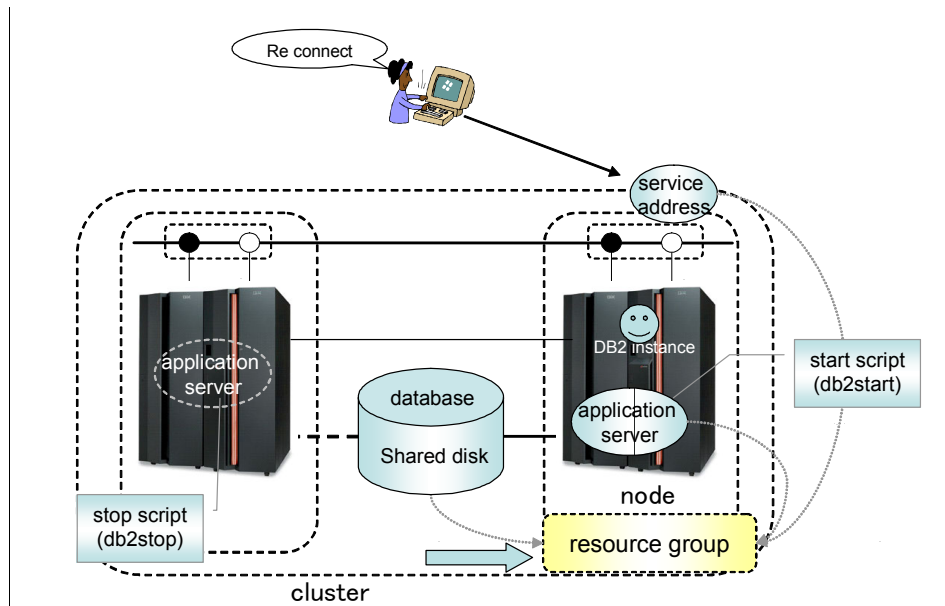


Figure 8-3 Switch over (planned takeover for maintenance)

## 8.3 Planning the HACMP cluster

When you plan the HACMP cluster, you can choose the configurations that best fit your system environment:

### ► Active/standby:

The typical configuration is active/standby configuration. In this configuration one node is active and the other is on standby. One resource group is shared between the nodes. This configuration is very simple and easy to manage but one node is not utilized for service.

### ► Active/active:

The active/active is also referred to as mutual takeover. In an active/active configuration, a database is usually running under each node. Figure 8-4 illustrates the mutual takeover configuration. Database\_A is running on node\_A and database\_B is running on node\_B. If node\_A fails, the resource group for database\_A is taken over by node\_B. Both database\_A and database\_B are then running on one node: node\_B. If node\_B crashes, the same operation happens, just in the opposite direction.

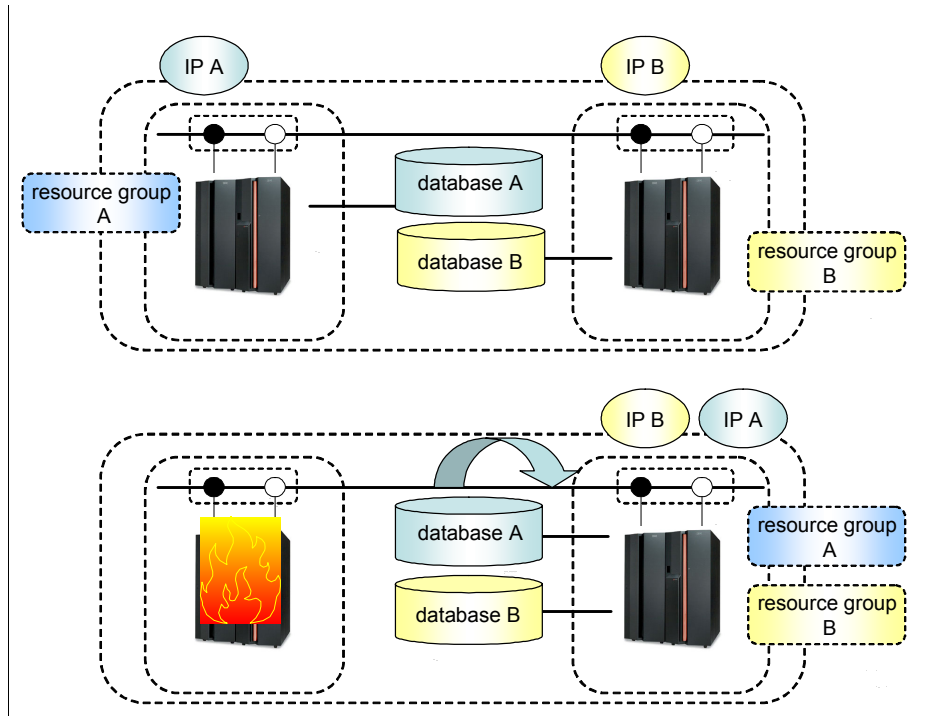


Figure 8-4 Mutual takeover

This configuration is called active/active since each node has its own database running and both are active at the same time. In the meantime, both nodes play the standby role for the other one and can takeover each other's resources. This is why this configuration is also called a mutual takeover configuration. With this configuration, you can exploit both machine resources for services, but this is more complicated compared to active/standby.

For this configuration, you need to plan carefully so that each node has enough machine resources (CPU or physical memory) to carry two databases while running. Let us say you have 1GB physical memory on each node and have allocated 600MB for a DB2 buffer pool in each database. After node\_A fails, database\_A is moved to node\_B. Total memory requirement now exceeds what node\_B has. What happens next is a huge page out to paging space which sometimes causes the system to hang, and at the very least impacts performance to a degree where no work is possible.

► Other variations:

There are some variations of cluster configurations for more than three nodes in a cluster. See Figure 8-5.

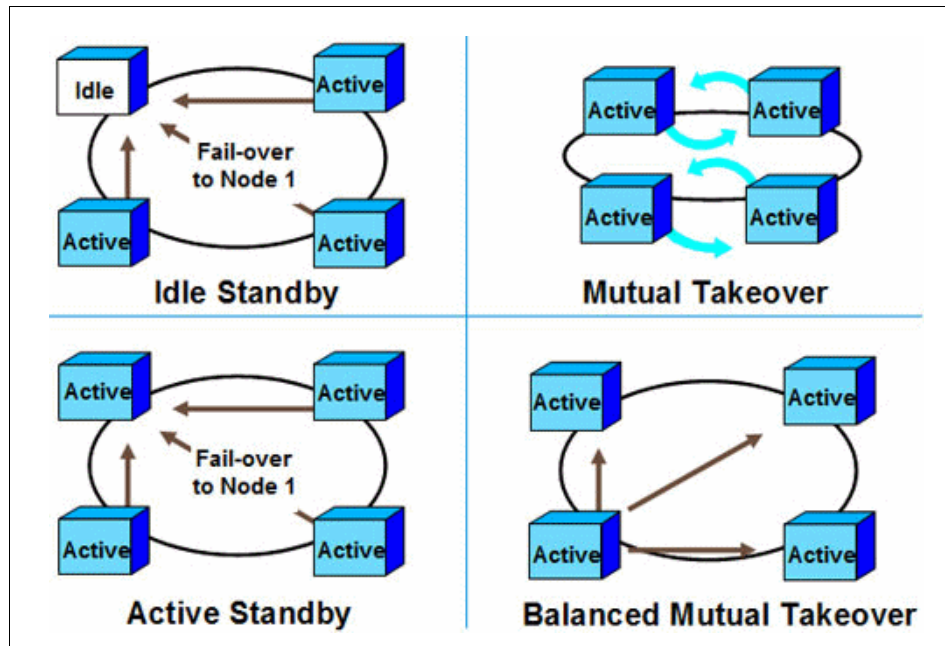


Figure 8-5 Variations of cluster configuration for multiple nodes

For more details about the DB2 highly available data store, refer to the IBM white paper “DB2 Universal Database and the Highly Available Data Store” at:

<http://www.ibm.com/developerworks/db2/library/techarticle/0310me1nyk/0310me1nyk.html>

## 8.4 Setting up HACMP

Setting up HACMP usually requires the following steps:

1. Define the cluster and add nodes to the cluster.
2. Configure the HACMP network and decide which method is to be used for IP address takeover.
3. Configure the communication interface and devices. A heart-beating disk can be used as the heart beat device. To configure the heart-beating disk, the shared disk should be in an enhanced concurrent volume group.

4. Configure the high availability service IP.
5. Configure the application server.

It is often useful to place a simple test script (dummy scripts) for the corresponding start and stop script entry to test the basic functions of HACMP. After verifying that the HACMP environment functions properly, modify the start and stop script to work.

6. Configure the High Availability (HA) resource group.

This will associate the HA service IP, shared volume groups and files systems, the application server and all the resources required by the application into one resource group. In case a node failure happens, the HACMP will move the resource group running on the failed node to the surviving node in the cluster.

7. Verify and synchronize HACMP configuration.

After configuring HACMP, you must verify and synchronize the topology and resource group using the facility provided by HACMP (for example, via SMIT command panels). Every time the HACMP configuration is changed, you must re-synchronize the cluster from the node where the change has been made.

After the verification and synchronization is successful, you may start the HACMP service.

### 8.4.1 HACMP cluster setup planning

Before you set up an HACMP environment, you need to plan the cluster environment. Consider the following list of items:

- ▶ Physical nodes
- ▶ Network
- ▶ Application back end/services software (for example, DB2)
- ▶ HACMP configuration
- ▶ Application server start/stop scripts

Next, we examine the planning of these items briefly in a sample environment specific to a disk sharing configuration. Detailed information and instructions for each item can be found in the links provided in 8.4.2, “HACMP configuration” on page 193.

## Sample environment

Figure 8-6 shows the sample configuration of an active/standby HACMP cluster configuration, with a normal DB2 instance in a shared disk configuration.

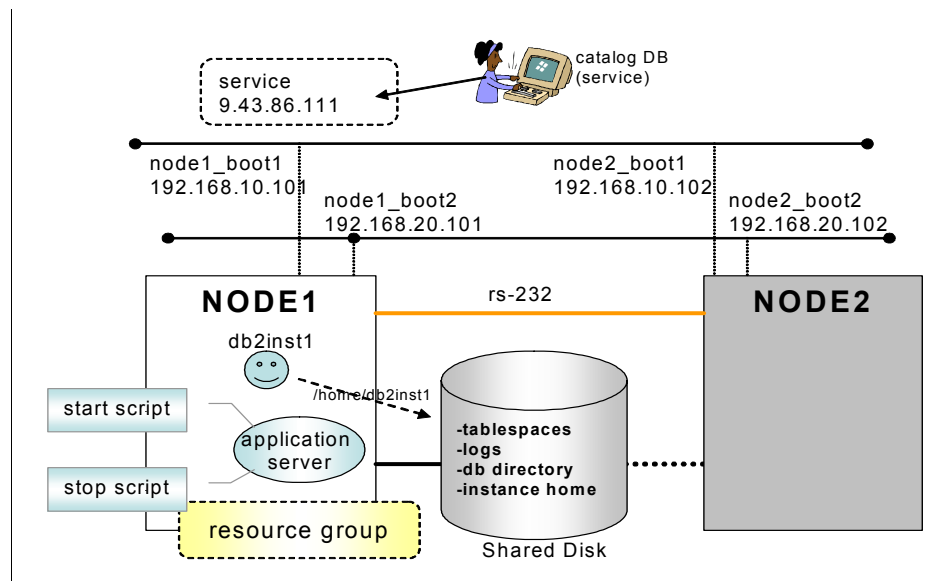


Figure 8-6 HACMP disk shared cluster

Here is the planning we did for this sample environment:

- Physical node configuration:
  - Two physical nodes named Node1 and Node2 are defined in the cluster:
    - Node1 is designated as the *service* node normally providing service for clients. A single partitioned DB2 instance is running on this node.
    - Node2 is the *standby* node which normally stands by for any service node outage. No DB2 instance runs on the standby node. The role of both nodes changes in response to system failover, or to planned takeover issued by administrators.
- The network configuration is set up as follows:
  - Two ethernet network interfaces on each node are provided for client connectivity. These are both under the control of HACMP. The service address for clients is added on one of these network interfaces.
  - One Serial (RS-232C) network is configured for HACMP keepalive. A serial network (non TCP/IP network) is recommended, as it makes HACMP failure detection more reliable.

► DB2 configuration:

The DB2 product libraries are installed on the local disk of each server. A single partitioned instance named *db2inst1* and its database named SAMPLE are created on a shared disk. If table spaces or logs are placed in different devices within the shared disk, those devices would all have to be included in a single resource group.

► HACMP configuration:

- A resource group named *shared\_rg* is configured in the cluster, which has a service IP address, application server, and file systems on a shared disk as resources.
- A service address is defined in the resource group. Each DB2 client has the entry for this service address in its catalog node directory and connects to the Service node through this IP address.
- An application server is also defined in resource group. The *application server start script* simply starts up database instance and restarts the database. The *application server stop script* stops instance.

## 8.4.2 HACMP configuration

In the last section, we briefly discussed the planning stage prior to HACMP implementation. In this section we introduce an outline of actually setting up HACMP in a shared disk cluster configuration. For more details and a step-by-step guide to set up HACMP and DB2, refer to these documents:

- *IBMR DB2 Universal Database Enterprise Edition for AIX and HACMP/ES*  
<ftp://ftp.software.ibm.com/software/data/pubs/papers/db2ee-aixhacmp.pdf>
- *DB2 UDB V8.1 ESE and high availability on AIX with HACMP 4.4.1: A mutual takeover scenario*  
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0407nikolopoulou/>

### Checking the network connection

Before configuring HACMP, the network must be configured properly. The following steps show how to check network configurations for the cluster.

1. Check that IP addresses are configured on network interfaces on both nodes.
2. Check that the */etc/hosts* file has all entries of IP addresses and their labels are those used by HACMP.
3. Verify that the name resolution is working well using the **host** command. If something is wrong, check and modify the */etc/hosts* file.

4. Check the serial network connection. The subsidiary network for HACMP keepalive is recommended to make HACMP failure detection more secure. For more detail, see *HACMP Administration Guide*, SC23-4862-09.

## Configuring the shared disk

This requirement is specific to an HACMP shared disk configuration. Because shared disks are such an integral part of the HACMP setup, this section lists the high level steps needed to set up shared disk resources.

Following are some of the terms concerning AIX Logical Volume Manager, which are used in this section:

- ▶ Volume group (VG)
- ▶ Logical volume (LV)
- ▶ Journaled file system (JFS)
- ▶ File system (FS)
- ▶ Log that maintains a consistent JFS (JFSLog)

Perform these steps to set up shared disk drives and the logical volume manager:

1. Check the disk drives.

Check that the external disk is configured and recognized from both nodes.

2. Create the VG.

Create the VG on the service node. The VG must have a unique name and major number (serial ID of VG) for all nodes in the cluster. You can check available major numbers on both nodes with the **lvfstmajor** command:

```
root@node1:/# lvfstmajor
58...
```

To add a volume group you can use the **smit** menu:

```
#smit vg
```

From **Volume Groups** → **Add a Volume Group** → **Add an Original Volumes Group**. Enter *VOLUME GROUP name*, and the available *Volume group MAJOR NUMBER*. as shown in Example 8-1.



### Example 8-1 Add volume group

---

#### Add an Original Volume Group

---

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
VOLUME GROUP name	[db2vg]	
Physical partition SIZE in megabytes		+
* PHYSICAL VOLUME names	[hdisk1]	+
Force the creation of a volume group?	no	+
Activate volume group AUTOMATICALLY at system restart?	yes	+
Volume Group MAJOR NUMBER	[]	+#
Create VG Concurrent Capable?	no	+

---

Activate the newly created VG by issuing the following command:

```
# varyonvg db2vg
```

#### 3. Create the JFSLog.

Create the JFSLog with a unique name on the new VG. When creating the first file system on a new VG, AIX will automatically create a JFSLog, with the name of loglv00, loglv01, and so on, for each new JFSLog on the machine. By default, AIX creates only one JFSLog per VG. Because a unique name is needed for the JFSLog, it is best to define the JFSLog with the **mk1v** command, before creating the first file system.

To create a JFSLog with the name *db2vgjfslog* in the VG *db2vg*, issue the following command:

```
# mk1v -t jfslog -y db2vgjfslog db2vg 1
```

To format the JFSLog, issue the following command, and select y when asked whether to destroy the LV:

```
# logform /dev/db2vgjfslog
```

```
logform: destroy /dev/db2vgjfslog (y)? y
```

#### 4. Create a file system.

Create any LVs and JFSs that are needed, and ensure that they have unique names and are not currently defined on any node. Set the FSs so that they are not mounted on restart. Verify the current LV and JFS names.

To create the LV for the */home/db2inst1* file system, issue the following command:

```
# smit lv
```

In the Add a Logical Volume menu, select a VG name and press **Enter**. The Add a logical Volume menu will be presented. Fill in the fields as shown in Example 8-2.

*Example 8-2 Add logical volume*

Add a Logical Volume		
Type or select values in entry fields. Press Enter AFTER making all desired changes.		
[TOP]	[Entry Fields]	
Logical volume NAME	[homelv01]	
* VOLUME GROUP name	db2vg	
* Number of LOGICAL PARTITIONS	[16]	#
PHYSICAL VOLUME names	[]	+
Logical volume TYPE	[]	+
POSITION on physical volume	middle	+
RANGE of physical volumes	minimum	+
MAXIMUM NUMBER of PHYSICAL VOLUMES to use for allocation	[]	#
Number of COPIES of each logical partition	1	+
Mirror Write Consistency?	active	+
Allocate each logical partition copy on a SEPARATE physical volume?	yes	+
RELOCATE the logical volume during reorganization?	yes	+
[MORE...9]		

Once the LV has been created, we can create a file system associated with this LV. In this example, we create a JFS2 file system. To add a JFS2 file system on a previously defined LV, issue the following command:

```
# smitty jfs2
```

Select **Add an Enhanced Journaled File System on a Previously Defined Logical Volume** and fill in the fields. See Example 8-3.

*Example 8-3 Adding a JFS2 file system*

---

Add an Enhanced Journaled File System

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
* LOGICAL VOLUME name	home1v01	+
* MOUNT POINT	[/home/db2inst1]	
Mount AUTOMATICALLY at system restart?	no	+
PERMISSIONS	read/write	+
Mount OPTIONS	[]	+
Block Size (bytes)	4096	+
Logical Volume for Log		+
Inline Log size (MBytes)	[]	#
Extended Attribute Format	Version 1	+
ENABLE Quota Management?	no	+

---

After a file system is created, it is not automatically mounted. Check that the file system can be mounted manually with the following command:

**#mount /home/db2inst1**

5. Unmount all of the FSs and deactivate the VG.

To do this, invoke the following commands:

**#unmount /home/db2inst1**

**#varyoffvg db2vg**

6. Import the VG.

Import the VG on the standby node (node2) with the same major number, and change the VG so that it is not activated on restart. When the VG is imported on the node2, the definitions of the FSs and LVs will be imported to node2. If we ever need to NFS export the file system, the major number for the VG has to be identical on both nodes. Make sure that the VG is defined to not be activated automatically on reboot, because it can be activated and controlled by HACMP.

To import a VG, issue the following command:

**# smit vg**

Select **Import a Volume Group** and fill in the fields. See Example 8-4.

Import a Volume Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]	
VOLUME GROUP name	[db2vg]	
* PHYSICAL VOLUME name	[hdisk1]	+
Volume group MAJOR NUMBER	[58]	+#

Next, change the VG so that it is not activated on reboot:

```
# smit vg
```

Select **Set Characteristics of a Volume Group** → **Change a Volume Group**. Then select **VOLUME GROUP name**. In the Change a Volume Group menu, select **no** on Activate volume group AUTOMATICALLY.

7. Move the VG back to the service node.

Move the file system and VG back to node1 for the next step where we will create a DB2 instance user on node1.

```
# umount /home/db2inst1
# varyoffvg db2vg
```

Next, run the following on node1:

```
# varyonvg db2vg
# mount /home/db2inst1
```

User/group setup and DB2 installation

Now that the shared disk is set up, we can create the DB2 instance and database on the shared disk. On node1, perform the following the steps:

1. Create the group for the DB2 instance.  
Use the **mkgroup** command to create the group.
2. Create the user for the DB2 instance.  
Use the **mkuser** and **passwd** commands.
3. Change the ownership of the FSs and LVs.
4. Install DB2 and set up the license key. The DB2 product has to be installed on the local disk of both nodes.
5. Create the DB2 instance.
6. Create a database in the shared file system.

7. Fail instance home directory over to node2.

Unmount all the FSs on the shared disk and varyoff the VG on node1. Activate the VG and mount all the FSs on node2. Delete the directory /home/db2inst1/sqllib to allow us to create the instance and control files on node2.

Next, on node2, repeat steps 1 to 6.

## Preparing application server scripts

The application server is composed of a start script and a stop script. In this section we explain what has to be included in these application server scripts.

- ▶ In the start scripts, we simply restart the DB2 instance and activate the databases to run crash recovery.
  - DB2 Enterprise Server Edition (ESE) instance has the *db2nodes.cfg* file in the sqllib subdirectory of the instance home. This file must contain the correct host name and IP address when you restart the DB2 instance on the other node. DB2 single partition instances also have this file. We explain this topic in more detail in 8.5, “Considerations for db2nodes.cfg file” on page 203.
  - Set the database configuration parameter AUTORESTART to ON. The database manager automatically calls the restart database utility, if needed, when an application connects to a database or a database is activated. The default setting is ON.
- ▶ In the stop scripts, we have to stop all applications and processes which access the shared disk to ensure that the node can release the shared disk successfully. In DB2 terms, this means **force application all, deactivate** for all databases, **terminate** to stop the back end process, and **db2stop** to stop the dbm/instance. Escalation commands such as **db2stop force** and **db2\_kill** may be necessary in order to get applications disconnected in a reasonable period of time.

A sample script file is packaged with the DB2 ESE for AIX to assist in configuring for HACMP failover or recovery in either hot standby or mutual takeover nodes. The script file is called *rc.db2pe.ee* for a single node (other than ESE) and *rc.db2pe.eee* for multiple nodes. They are located in the sqllib/samples/hacmp/es subdirectory of the DB2 Instance home. The appropriate file can be copied and customized for your system. Once customized and renamed, rename *rc.db2pe.ee* to *rc.db2pe*. For example, these sample scripts are designed to be called with syntax **rc.db2pe db2inst1 start**, and **rc.db2pe db2inst1 stop**, respectively.

For more information about HACMP/ES events, refer to the section on HACMP ES Event Monitoring and User-defined Events in the Cluster Support for AIX chapter of the DB2 9 manual *Data Recovery and High Availability Guide and Reference*, SC10-4228. This topic is also available in the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/c0007500.htm>

## Configuring HACMP

AIX **smitty** provides HACMP configuration interfaces. The main steps are as follows:

1. Add nodes to the HACMP cluster
2. Add a service IP label/address
3. Configure application servers
4. Add a resource group
5. Add resources to the resource group
6. Define serial network and serial network device
7. Verify and synchronize HACMP configurations
8. Basic verification of HACMP configuration

Here we expand upon each step:

1. Add nodes to HACMP cluster.

```
#smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Add Nodes to an HACMP Cluster**

In Configure Nodes to an HACMP Cluster (standard) menu, enter the cluster name and new nodes.

2. Add a service IP address.

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Service IP Labels/Addresses** → **Add a Service IP Label/Address**

In the Add a Service IP Label/Address menu, enter the IP label and network name

3. Configure application servers.

To access this menu, proceed as follows:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration → Configure Resources to Make Highly Available → Configure Application Servers → Add an Application Server**

In the Add application Server menu, enter the server name, and complete paths of the start and stop scripts.

**Note:** The start and stop scripts that are called from the application server have to exist in the local directory on both nodes and have the same name.

4. Add a resource group.

To add a resource group, go to **smitty**:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration → Configure HACMP Resource Groups → Add a Resource Group**

In the Add a Resource Group menu, enter the resource group name and participating node names.

5. Add resources to the resource group:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration → Configure HACMP Resource Groups → Change/Show Resources for a Resource Group (standard)**

In the Change/Show Resources for a Resource Group menu (Example 8-5), enter the service IP label/address, application server name, volume groups, and file system information.

The resource group policies we set in this example are as follows. This corresponds to the *Rotating resource group* in former HACMP versions, which means there is no priority for resource groups between nodes. For further details, see Chapter 6 in *HACMP v5.3 Planning and Installation Guide*, SC23-4861-07.

### Example 8-5 Adding a resource to a resource group

Change/Show All Resources and Attributes for a Resource Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

	[Entry Fields]
Resource Group Name	db2_rg
Participating Nodes (Default Node Priority)	node1_bt node2_bt
Startup Policy	Online Using Distribution Policy
Fallover Policy	Fallover To Next Priority Node In The List
Fallback Policy	Never Fallback>
Service IP Labels/Addresses	[service] +
Application Servers	[db2_server] +
Volume Groups	[db2vg] +
Use forced varyon of volume groups, if necessary	false +
Filesystems (empty is ALL for VGs specified)	[/home/db2inst1 ] +

#### 6. Define the serial network and serial network device.

If you have a serial network as the subsidiary keepalive network, you can configure this from the **smit** menu.

#### 7. Verify and synchronize HACMP configurations.

After defining the HACMP configuration from one node, you must verify and synchronize the cluster topology to the other node.

**# smitty hacmp**

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Verify and Synchronize HACMP Configuration**.

HACMP verification utility checks that the cluster definitions are the same on all nodes and provides diagnostic messages if errors are found.

The HACMP configuration details can be found in the following documents:

- ▶ *High Availability Cluster Multi-Processing for AIX* PDF manual links:  
[http://www-03.ibm.com/systems/p/library/hacmp\\_docs.html](http://www-03.ibm.com/systems/p/library/hacmp_docs.html)
- ▶ *HACMP v5.4 Administration Guide*, SC23-4862-09
- ▶ *HACMP v5.3 Administration Guide*, SC23-4862-06
- ▶ *HACMP v5.3 Planning and Installation Guide* SC23-4861-07
- ▶ *Data Recovery and High Availability Guide and Reference*, SC10-4228



## 8.5 Considerations for db2nodes.cfg file

From version 8 onwards, DB2 ESE single and partitioned instances have a db2nodes.cfg file in the sqllib subdirectory of the Instance home. The db2nodes.cfg file is used to define the database partition servers that participate in a DB2 instance. In the cluster environment, you have to consider the entry of this file to start DB2 instances on different nodes.

Suppose that a DB2 ESE single partition instance is running in a cluster which is composed of a service node named node1 and a standby node named node2. When a takeover occurs, the DB2 instance has to be started on node2. If the Instance home directory is configured on the shared disk, the entry in db2nodes.cfg file on the shared directory will not match the host name of the standby node. Then the **db2start** command fails with error codes like -6048 or -6031 when the application start script tries to start the DB2 instance. Example 8-6 shows the case of **db2start** failing when db2nodes.cfg does not match the host name.

*Example 8-6 Error messages caused by an invalid db2nodes.cfg entry*

---

```
$hostname
host2

$cat /home/db2inst1/sqllib/db2nodes.cfg
0 host1 0

$ db2start
SQL6048N A communication error occurred during START or STOP DATABASE
MANAGER processing.
```

---

You have several options to avoid this error, some of which are listed below:

1. Modify the file entry in the start script.
2. Use the **db2start** command with the restart option.
3. Use the **db2gcf** command with the -u option.
4. Use an alias in the hosts files.
5. Use the service IP address in db2nodes.cfg.

Options 1, 2, and 3 modify the entry of db2nodes.cfg file manually or automatically, while options 4 and 5 do not. Options 2 and 5 require permission for remote execution, while options 1, 3, and 4 do not.

Next we describe each of these options in further detail.

## Modifying the file entry in the start script

Perhaps the simplest of the listed methods, this entails modifying the entry of the db2nodes.cfg file before starting the DB2 instance, or preparing another configuration file with the correct entry and overwriting the existing db2nodes.cfg file. In our example, a DB2 instance, on the HACMP Service node named node1, (node2 as standby), has a db2nodes.cfg file shown below:

```
0 node1 0
```

In the event of a takeover on node2, the db2nodes.cfg file shown above would be invalid, so the application start script can include a process to modify db2nodes.cfg as below, or prepare a local file which already has the entry below and copies or overwrites this file to the db2nodes.cfg file before actually starting the DB2 instance on node2.

```
0 node2 0
```

Figure 8-7 shows this concept graphically, with states shown before, during, and after the takeover by node 2 has occurred.

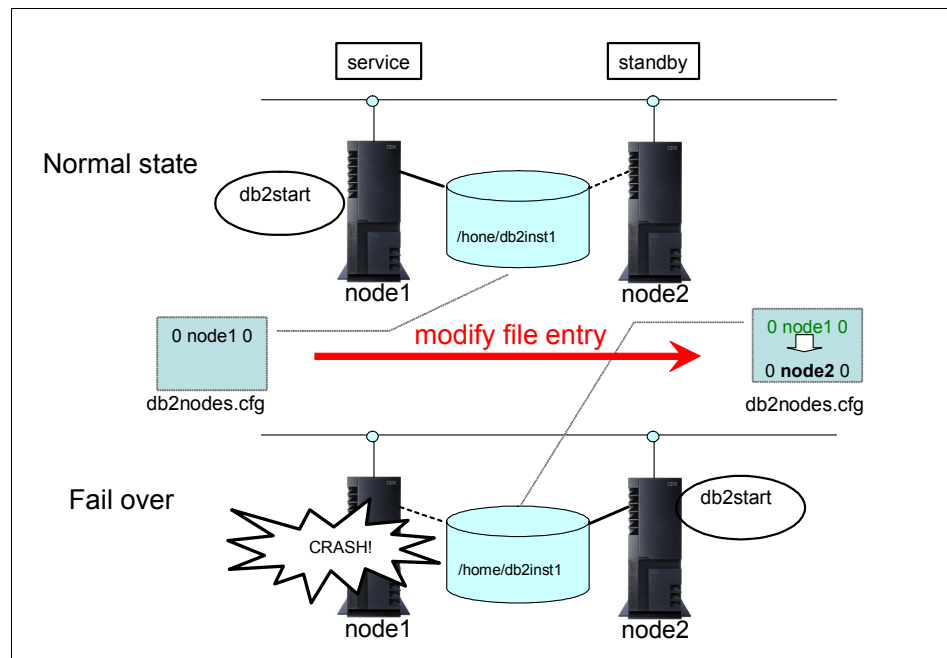


Figure 8-7 Modifying the db2nodes.cfg entry before starting DB2 on node2

## Using the db2start command with the restart option

Starting the DB2 instance with the restart option gives the ability to specify the new host name as the node that is taking over the resource groups. See Example 8-7. Note that DB2 automatically modifies the db2nodes.cfg file with the host name you specified in the restart option.

*Example 8-7 Restart option of the db2start command*

---

```
db2start dbpartitionnum 0 restart hostname node2
```

---

### Considerations

Since the restart option requires the permission of remote execution, you have to configure remote shell (rsh) or secure shell (ssh) to be available in the cluster. This is required for both single and multiple node partitions.

► rsh configuration:

Add entries of a reliable host name and user to the .rhosts file or hosts.equiv file. This option is often not viewed favorably due to the security risks which remote shell may present.

► ssh configuration (available from DB2 V8.2):

Install ssh, set the public key and private key for the use of the DB2 instance. Set full path of **ssh** command in DB2RSHCMD registry value as follows:

```
db2set DB2RSHCMD=/usr/bin/ssh
```

## Using the db2gcf command with the -u option

This option uses the **db2gcf** command. When issuing the **db2gcf** command with the “-u” option, the db2nodes.cfg file is automatically modified and the DB2 instance starts on the standby node. With this method, rsh or ssh are not required, so the method is useful if you have a site or company policy against using any .rhosts files for security reasons, or if you cannot have ssh installed. See Example 8-8 for a **db2gcf** command example and Example 8-9 for the db2nodes.cfg file modified by this command. You may notice in Example 8-9 that a fourth parameter has been appended to db2nodes.cfg. This is *Netname*, used for interconnect for FCM communication. For more details, refer to *Quick Beginnings for DB2 Servers*, GC10-4246.

*Example 8-8 db2gcf command issued with the -u option*

---

```
$ db2gcf -u -p 0 -i db2inst1
```

```
Instance   : db2inst1
```

```
DB2 Start  : Success
```

```
Partition 0 : Success
```

---

*Example 8-9 Entry of db2nodes.cfg file modified by the db2gcf command*

---

```
0 node2 0 node2
```

---

For more details on db2gcf, refer to *Command Reference*, SC10-4226, or the DB2 Information Center at:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.admin.doc/doc/r0010986.htm>

## Using an alias in the hosts file

This option uses an alias in the `/etc/hosts` file. The entry of the `db2nodes.cfg` file never changes during takeover. With this method, `rsh` or `ssh` are *not required*, so the method is useful if you have a site or company policy against using any `.rhosts` files for security reasons, or if you cannot have `ssh` installed.

You have to create an alias entry in the `/etc/hosts` file on both systems with the following format:

```
ip_address short_name long_name alias
```

On node1 (primary system), the entry looks similar to Example 8-10.

*Example 8-10 /etc/hosts file of node1*

---

```
192.168.10.101    node1    node1.itsosj.sanjose.ibm.com    db2host
```

---

On node2 (standby system), the entry looks similar to Example 8-11.

*Example 8-11 /etc/hosts file on node2*

---

```
192.168.10.102    node2    node2.itsosj.sanjose.ibm.com    db2host
```

---

The IP address and domain name for both files should match network definitions for the primary and standby systems, that is, the alias goes against the actual local host name entry, as DB2 uses the host name as the basis for what should be in `db2nodes.cfg` before it will start.

Next, the hosts entry in the `/etc/netsvc.conf` file must contain `local` as the first parameter. The hosts entry on both primary and standby systems would then look similar to Example 8-12.

*Example 8-12 /Entry of etc/netsvc.conf file*

---

```
hosts=local,bind
```

---

By putting local as the first parameter, this will force the system to look in the /etc/hosts file for a host name entry before going to the Domain Name Server (DNS).

Finally, parameter/column two in the db2nodes.cfg file is changed to the alias defined on both systems. The new db2nodes.cfg file then resembles Example 8-13, where db2host is the name defined in the hosts file on both systems.

*Example 8-13 Entry of db2nodes.cfg file*

---

```
0 db2host 0
```

---

After completing these steps, fallover and fallback testing should be done to ensure the HA computing environment is working correctly.

## Using service address

This option makes use of remote execution. It either requires the creation of a .rhosts file in the DB2 instance home directory, or ssh to be properly configured.

► **.rhosts option:**

The following network interface entries must be included in this .rhosts file:

- Primary system host name
- Standby system host name
- Service adapter name as defined in the HACMP configuration

If *node1* is the host name of the primary system, *node2* is the host name of the standby system, and *service* is the service adapter name for the cluster, the .rhost file would look like Example 8-14 where db2inst1 is the DB2 instance name.

*Example 8-14 Entry of .rhosts file*

---

```
node1      db2inst1
node2      db2inst1
service    db2inst1
```

---

► **ssh configuration (available from DB2 V8.2.):**

Install ssh, and set the public key and private key for the use of the DB2 instance. Set the full path of the **ssh** command in DB2RSHCMD registry value as follows:

```
db2set DB2RSHCMD=/usr/bin/ssh
```

The next step is common to both options:

- Parameter/column two in the db2nodes.cfg file must be changed to the service adapter name of the HACMP configuration. Example 8-15 shows db2nodes.cfg file content, where *service* is the service adapter name.

*Example 8-15 Entry of db2nodes.cfg file*

---

```
0 service 0
```

---

After completing these steps, failover and fallback testing should be done to ensure the HA computing environment is working correctly.

## 8.6 Tuning tips for quick failover

When an outage on a primary database occurs, the steps required for continuing database services are as follows:

1. Failure detection
2. Resource failover required to provide service
3. Restart applications and services

To speed up the recovery time, we need to reduce the time taken in each step. In this section we discuss the considerations for each step.

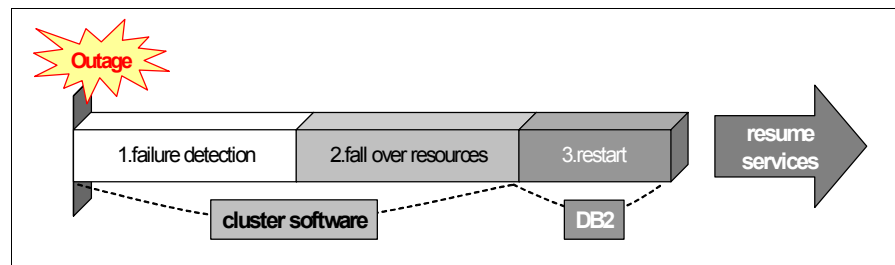


Figure 8-8 Time consumed by failover process

### 8.6.1 Failure detection time

Failure detection time can be tuned by the cluster software settings. For HACMP, *failure detection rate* for each network type can be specified. There are several network types such as: atm, ethernet, fddi, hps, token, diskhb, rs232, tmcsai, tmssa. Having the same detection rate is recommended through all networks in the cluster.

HACMP provides three default settings: SLOW(48s), NORMAL(20s), and FAST(10s). You can also specify a custom failure detection rate in seconds.

Note that setting a very low number of seconds can cause misjudgment of failure detection (false positives), resulting in unnecessary frequent failover.

## 8.6.2 Failover the resources

In this section, we describe several options to speed up resource failover time.

### IP address takeover

In previous versions of HACMP, network interfaces were directly overwritten by the service address. From HACMP/ES V4.5, an IP alias takeover is available, which reduces time for adding the service IP address to the standby node. As stated in the *HACMP v5.3 Administration Guide*, SC23-4862-06, the reduced time is due to fewer commands being required when moving addresses.

For more information on IP Aliases in HACMP, see the HACMP manuals:

- ▶ *HACMP v5.3 Administration Guide*, SC23-4862-06
- ▶ *HACMP v5.3 Planning and Installation Guide* SC23-4861-07

### Disk resource failover

During a failover, HACMP moves shared disks defined in the resource group from one server to another. This includes the following activities:

- ▶ *Varyon* (activate) VG - **varyonvg** command
- ▶ Mount FSs (if the FSs are defined in the resource group)

The delay in a DB2 failover mostly stems from the need to move control of the shared disks and resources from one server to another. An integrity check of file systems (**fsck**) consumes a tremendous amount of time when the shared VG contains huge FSs. The more FSs in the resource group, the longer it takes to failover. Therefore, the basic way to speed up disk resource takeover is to reduce the amount of FSs in the resource group. This can be accomplished by using raw device containers instead of FSs. With DB2 database managed space (DMS) on raw device, we can significantly decrease the disk resource failover time, because there is no need to mount and check consistency of FSs during failover.

We can also have a *concurrent access resource group* for the raw disk, reducing failover time even further. *Concurrent access volume group* with HACMP enables both nodes to activate a VG simultaneously. This configuration reduces time for takeover resources, since a shared disk is then always activated on standby nodes and there is no need to activate/varyon the VGs either. Using concurrent access with HACMP requires the installation of an additional HACMP file set. For example:

```
cluster.es.clvm.rte for 5.3.0.0 COMMITTED ES for AIX Concurrent
Access
```

Concurrent access mode is not supported for JFSs. Instead, you must use only raw logical volumes or physical disks for concurrent access resource groups.

Another consideration for the concurrent access resource group concerns the logistics of ensuring that no processes on the standby node (for example, DB2 instances) are actively using any shared disks. Concurrent VGs are activated from all the nodes in the cluster. This means that they can be accessible from all nodes in the cluster simultaneously and there is a very real possibility of unintentional data corruption or loss. In this situation, software with data on a shared disk has the responsibility to ensure consistent data access, whether that be performed from within the software, or controlled externally via the application start/stop scripts.

While DB2 databases will not tolerate concurrent access from multiple servers, in mixed/hybrid HACMP/HADR mode, as discussed in Chapter 10, “HADR with clustering software” on page 253, DB2 databases use a shared disk on the primary cluster pair, and HADR writes to the standby on a third server node. Of course, only one DB2 instance can actively use this shared disk at once.

To improve performance of failover/takeover, using raw device for data storage is recommended as mentioned above. Then the next question is a matter of what is created on the raw device and what is not.

- ▶ User table spaces:

If your database is huge, user table spaces should take up most part of the shared disks, so using DMS raw device for the table space container will improve failover performance.

- ▶ Temporary table spaces:

You can configure temporary table spaces in raw devices too. For huge sequential read and write I/Os, DMS temporary table space is preferable in terms of performance. On the other hand, SMS table space is recommended from the point of view of storage utilization and ease of manageability. One concession to help with DMS containers is the automatic table space resizing feature for table spaces available from DB2 V8 FixPak 9. For details of the automatic table space resizing, see DB2 InfoCenter:

<http://publib.boulder.ibm.com/infocenter/db2luw/v8/index.jsp?topic=/com.ibm.db2.udb.doc/admin/c0012277.htm>

- ▶ Catalog table space:

You can specify the characteristics of the DB2 system catalog table space when you create databases with the CREATE DATABASE command. Since the DB2 SYSCATSPACE is small in most cases, it is unlikely to contribute to reduction of failover time.



► Active logs:

An active log can be configured in raw device. Note that from DB2 version 9, database logging using raw devices is deprecated. For details, see the DB2 Information Center topic “Database logging using raw devices is deprecated”:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp?topic=/com.ibm.db2.udb.rn.doc/doc/c0023086.htm>

► Other database components:

Other database components, such as database directory files, instance home and subdirectories, and archive log directories have to be created on FSs. Note that the archive logs also have the option to use TSM and avoid the issue altogether.

Since we still have FSs for some database components, HACMP fast disk takeover can help to failover these file system resources. This recent feature from HACMP V5.1 enables faster resource group takeover using AIX Enhanced Concurrent Volume Groups (ECM). Enhanced concurrent VG of AIX supports active and passive mode varyon, can be included in a non-concurrent resource group, and the fast disk takeover is set up automatically by the HACMP software.

For all shared VGs created in enhanced concurrent mode that contain FSs, HACMP will activate the fast disk takeover feature. When HACMP starts, all nodes in a resource group sharing the same enhanced volume group will varyon this VG in passive mode. When the resource group is brought online, the node that acquires the resources will varyon the VG in active mode. This reduces the time to varyon VGs, even though it still requires time to mount FSs. The prerequisites for this functionality are:

- HACMP V5.1 or later
- AIX 5L™ 5.2 or higher
- bos.clvm.5.2.0.11 or higher
- APAR IY44237

For more information about fast disk takeover, see *HACMP for AIX 5L V5.3 Planning and Installation Guide*, SC23-4861-06.

There is another HACMP option to speed up mounting FSs: parallel mount of FSs. This feature allows for specification of how FSs are mounted during disk resource takeover. Parallel mount can be faster than sequential (default). This option can be used if you do not have nested FSs.

You can also change the method of the file system integrity check from the default `fsck` to `logredo`. Although `logredo` may speed up mounting FSs, this option does not guarantee that all the errors in the FSs can be fixed if the node crashes in the middle of file system I/O.

## Starting and activating the database

After the resources are taken over by the standby node, the next thing to do is to re-enable the DB2 database for client use, which includes the following activities:

- ▶ Restart the DB2 instance (by the **db2start** command)
- ▶ Activate the database, including buffer pool activation and crash recovery

If crash recovery is required, it can consume some time before the database can be opened up for client access. From the perspective of high availability, it is imperative to reduce crash recovery time and restart the database more rapidly.

Crash recovery is the process by which the database is moved back to a consistent and usable state. This is done by rolling back incomplete transactions and completing committed transactions that were still in memory when the crash occurred. In other words, crash recovery updates the data in table space containers to match what is stored in any unprocessed log records. This means that time consumed by crash recovery depends on how large the gap is between the log records and the actual data in database containers. Here, the gap equates to the amount of log records as follows:

- ▶ Records to be rolled forward: The amount of log records that have been committed, but have not been written from the buffer pool to the disk.
- ▶ Records to be rolled back: The amount of log records that have not been committed, but have been written to disk.

To make crash recovery fast, you have to reduce these amounts:

- ▶ Records to be rolled forward:

You can synchronize memory and database pages more frequently by writing the dirty pages on buffer pool to the database pages. We introduce some relevant tuning parameters next.

### – NUM\_IOCLEANERS:

The page cleaner DB2 process (I/O cleaner) is in charge of writing dirty pages on the buffer pool to the database pages on disk. The number of these processes activated is determined by the database configuration parameter NUM\_IOCLEANERS. When you increase this parameter, the contents of the database will be updated more on disk, because the page cleaner cleans dirty pages from buffer pools to data pages and therefore this will reduce the crash recovery time. We recommend that you increase this parameter depending on the number of the CPU cores on the server.

– SOFTMAX and CHNGPGS\_THRESH:

The page cleaners are started according to the setting of the database configuration parameters SOFTMAX and CHNGPGS\_THRESH. Tuning these parameters lower than default values helps to reduce crash recovery time, with the trade-off of increasing the load on certain system resources.

When the SOFTMAX parameter is set lower, it will cause the database manager to trigger the page cleaners more often and take more frequent soft checkpoints. The log control file is then written to disk more often and thus can reduce crash recovery time.

The lower that the CHNGPGS\_THRESH parameter is set, the lower the percentage of changed pages is required to be kept in the buffer pool. This will trigger the asynchronous page cleaners to start cleaning the buffer pool more often. This also means it writes committed data to the disk more often and therefore less recovery time is needed in case of crash recovery.

While in theory, spreading out smaller but more frequent writes to disk from buffer pool pages should reduce peak write I/O load, and reduce the frequency of synchronous write I/O, frequent writes to storage by page cleaners may still sometimes lead to adverse performance impacts. Consider the performance impacts to system load and transaction throughput and tune for the best values to suit your system.

From DB2 V8.1.4 onwards, consider using registry variable DB2\_USE\_ALTERNATE\_PAGE\_CLEANING for even further improved and proactive page cleaner activity. This registry variable makes CHNGPGS\_THRESH redundant, as it is no longer controls page cleaning.

► Records to be rolled back:

It is important to frequently issue commit from the applications. This cannot be stressed enough; issues of transactional performance, concurrency, and ease of recoverability are heavily dependent on how applications are coded.

Despite giving the appearance of a continuous connection to a database back-end, a well coded online transaction will perform most of its work without even holding locks on data, and will only need to gain locks momentarily before immediately committing in order to process requests after the user has confirmed a given action.

Batch processing is also a frequently abused target from users who expect to be able to issue logged updates of millions of rows at a time, and then get upset when a logical failure or timeout occurs halfway through their input files, and their transaction then takes time to rollback. These are often the same users who complain when told that the answer is to commit as frequently as possible, meaning that they will have to code retry and restart logic into their batch jobs, or pursue alternatives such as unlogged loads with copy yes options.





## DB2 with Microsoft Windows Cluster Server

In this chapter we provide the steps to set up a highly available DB2 environment with Microsoft Windows Cluster. We explain the fundamental concepts of Microsoft Windows Cluster Server to help you understand its architecture and the basic configuration procedures in Windows 2003 Advanced Server.

We discuss how to configure a DB2 instance and other cluster resources to become the highly available resource in Window Cluster Server.

We demonstrate the configuration steps using both the automatic utility db2mscs and the manual steps.

**Note:** This chapter uses the screen shots from Microsoft Windows Cluster. Microsoft product screen shots are reprinted with permission from Microsoft Corporation.

## 9.1 Cluster Server concepts

Microsoft's answer to high availability requirements in business is Windows Cluster Server, formerly known as Microsoft Cluster Servers (MSCS).

MSCS was early introduced in the Windows NT® 4.0 Enterprise edition as a separate component from the operating system.

With the introduction of Windows 2003 Advanced Server and Data Center editions, MSCS has been revamped in what is now called Windows Cluster Server (which will be referred to as Cluster Server hereafter). Cluster Server is an integral part of the operating system and therefore cannot be uninstalled. It is always present (under %windir%\Cluster), therefore, no additional software pieces have to be installed.

The objective of this chapter is not to make an exhaustive description of Windows Cluster but to introduce the fundamental concepts so that the DB2 DBA can consider this feature with DB2 when selecting a clustering product for a DB2 application.

In this section, we introduce some concepts particular to Cluster Server. Some general concepts from cluster technologies are not mentioned as they have been described in Chapter 1, “DB2 high availability options” on page 3.

### 9.1.1 Types of clusters

Cluster Server provides technologies to create two types of highly available servers, depending on what services must be made available:

- ▶ Network Load Balance (NLB)
- ▶ Quorum cluster server

#### **Network Load Balance**

NLB is a type of cluster that allows enterprises to create a farm of servers (nodes) that provide a service to applications. In the event of a failure in one of the nodes, the workload of that node can be assimilated by the rest of the nodes in the farm. This type of cluster is suitable for applications that do not store a state. Applications that are usually capable to scale-out horizontally, such as Proxy servers, Web servers, and so on, are the target for this type of clustering solution.

An NLB server uses algorithms to balance the load of each node to generate the best possible overall performance. NLB is included in all versions of Windows 2003.

## Quorum cluster server

Quorum cluster servers can be used for applications that store a state (such as a Database Management System) and are used for services that usually scale vertically and have to be made highly available.

Since these services can scale-out vertically, a server farm cannot be created, and the only way to make the server highly available in case of node failure is to have a spare machine that will take over the functions of the failed node.

Cluster servers require the use of special shared disks among the nodes or the use of complex algorithms among nodes to synchronize the state of the nodes and the services they provide.

Cluster servers come in two different types:

- ▶ Single quorum servers
- ▶ Majority node set (MNS)

### ***Single quorum servers***

Single quorum servers use a shared disk device known as quorum. These shared disks are SCSI disks in a shared bus or a Storage Area Network (SAN) using Fiber Optics.

One of these shared disks is known as the *quorum* for the cluster. This quorum is used as an arbitrator to track which node is the owner of the resources and services. If the node that owns the resource goes down, Cluster Server takes action to move the service to another node.

### ***Majority node set***

MNS consists of two or more nodes that do not have a single shared quorum. Each node in the system has their own copy of the quorum in a local disk. The Cluster Server uses complicated algorithms to keep the local quorums synchronized. These nodes may or may not be attached to one or more cluster storage devices.

MNS clusters are more commonly seen in a geographically dispersed cluster configuration and that is the reason for not using shared devices. The down side of this type of configuration is that for the cluster to work, you require half of the nodes plus one online and interconnected.

This rule is necessary since all clusters are connected only throughout the network. The only way to know if a server has the correct quorum information is by knowing that the *majority* of the nodes have the same information.

## 9.1.2 Microsoft Cluster Server definitions

From now on we refer only to single quorum servers. In this section, we introduce a few concepts that are necessary to understand how single quorum servers work in Windows Cluster Servers.

### Resource type

In Cluster Server, a resource type is a service, such as Web service, IP address, or a device, such as a shared disk, that has to be made highly available.

For every type of resource to be manipulated, you require a handler which is an interface that will manipulate this resource. For example, if what you want to make highly available is a shared disk, you require a mechanism to:

- ▶ Detect that the resource is working properly.
- ▶ Detect the availability of the disk from each node.
- ▶ Failover or failback the resource among the nodes as necessary.

In the case of Cluster Server, this handler is implemented as a Dynamic Link Library (dll) that must be provided by the provider of the service or resource. In the case of basic operating systems or resources (such as disks), Cluster Server provides a default dll that can handle them.

In the case of DB2, a new type of resource known as DB2 should be created. The library that manages this resource is version dependent. The dll files and their location are as follows:

- ▶ DB2 Version 8: %DB2PATH%\db2wolf.dll
- ▶ DB2 Version 9: %windir%\MSCS\db2server.dll

### Resource

A resource is any element that you want to make highly available in the system. Common resources that you will find are:

- ▶ IP addresses
- ▶ DB2 instances
- ▶ File shares, which are shared directories in Windows
- ▶ Shared disks

### Groups

A cluster group is a set of resources that are interdependent of each other. They are logically linked together for a specific purpose, for example, to make a DB2 instance highly available.



Groups also make it possible to establish ownership of the resources to each node. When the group is created, you must specify what nodes can use the resources in that group and which node will be the initial owner.

## 9.2 Minimum configuration and sample setup

In order to set up a single quorum Cluster Server, you require at the very least:

- ▶ A Windows 2003 domain.
- ▶ Two machines to work as the nodes in the cluster using the Windows 2003 Advanced Server or Data Center Editions.
- ▶ A shared disk that can be accessed simultaneously from all the nodes.
- ▶ Two network cards in each node is desirable but not mandatory.

In our lab example, we have the following resources:

- ▶ We set up a machine Wisla, a Windows 2003 standard server, working as the Domain Controller for domain CHAVIN.
- ▶ Machines Mochica and Chimu running Windows 2002 Advanced Server.
- ▶ Each node has a QLogic QLA2200 PCI Fiber Channel adapter to access the SAN devices.
- ▶ Four disks are created in the SAN infrastructure for the nodes:
  - Drive Q: has 1 GB space and has been created to be the quorum of the cluster.
  - Driver R:, S: and T: are 50 GB partitions for DB2 table spaces and instance profile storage.

Following is the procedure to set up a highly available DB2 environment with Cluster Server. We provide detailed steps for creating Windows Clustering Server and configure DB2 with it in the next few sections. Refer to the Windows documentation for the steps mentioning the Windows domain controller.

1. Install Windows 2003 Advanced Server or Data Center Edition on each machine that will be a cluster node.
2. Configure the shared disk devices and make sure that each node has access to the shared resources.
3. Add the nodes to the Windows Domain.
4. Create a Cluster Server.
5. Add nodes to the Cluster Server.
6. Install DB2.

7. Create a DB2 instance.
8. Make the DB2 instance highly available.

## **9.3 Creating a cluster server**

In order to create a cluster, you require a Windows Domain. All the information related to the cluster is stored in the active directory of the domain. The machines that will be part of the cluster must use Windows 2003 Enterprise Edition or Windows 2003 Data Center Edition. The cluster services are not available in Windows 2003 standard edition.

We follow these steps to create a Cluster Server:

- ▶ Add the machines that will be the nodes to the domain.
- ▶ Create the cluster in the domain.

### **9.3.1 Adding the machines to the domain**

The first step is to attach the machines that will conform the cluster into the Windows domain. In order to accomplish this, follow this procedure for each machine.

1. Use Administrator user ID, go to **Start** → **Control Panel** → **System**. In System Property, select the **Computer Name** tab, then click **Change**. See Figure 9-1.

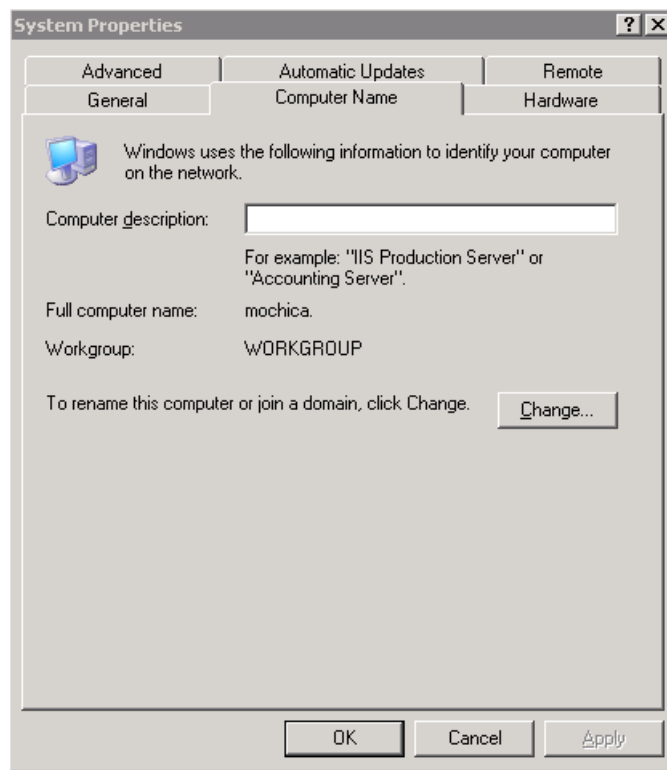


Figure 9-1 System Properties

2. In the Computer Name Changes window, select **Domain** and enter the Windows Domain name you want to join, CHAVIN in this case. Click **OK**. See Figure 9-2.

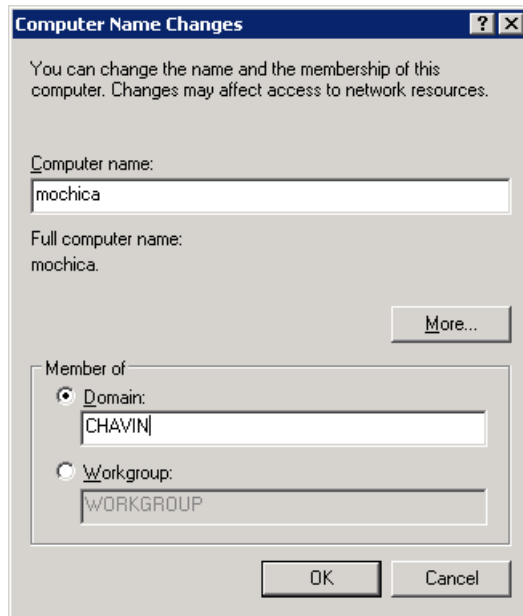


Figure 9-2 Add computer to the domain

3. A message will be shown indicating that the machine was accepted on the domain. See Figure 9-3.



Figure 9-3 Machine accepted message

### 9.3.2 Creating the cluster in the domain

The following steps are for creating a cluster in the domain. Remember that in order to create a cluster, a shared disk resource must exist. This shared disk must be connected to both nodes using a SCSI shared bus or a SAN adapter.

1. Go to **Start** → **Administrative Tools** → **Cluster Administrator**. The Cluster Administrator opens the Create New Cluster option (Figure 9-4). Click **OK**.

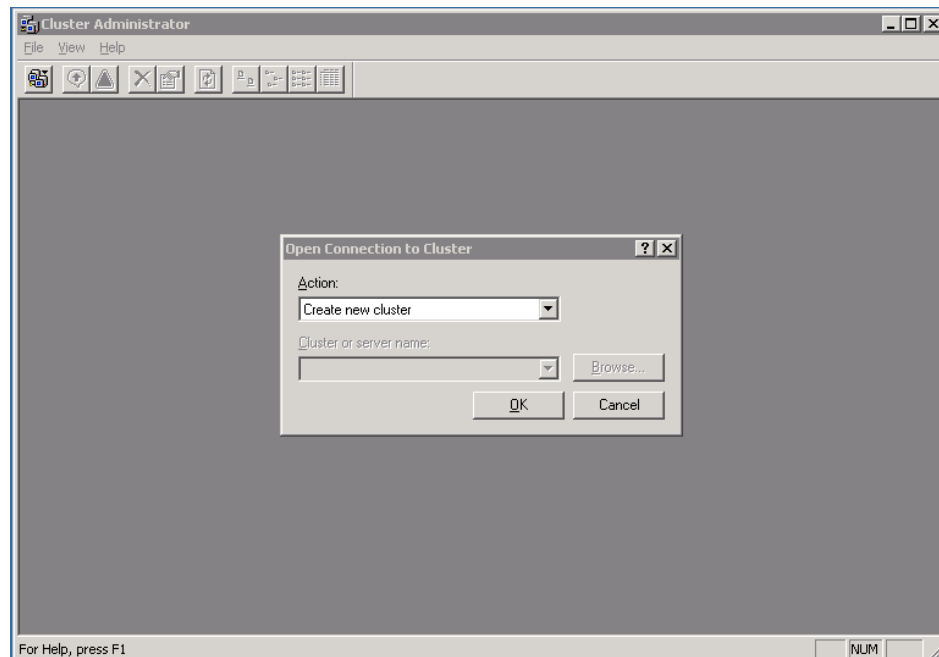


Figure 9-4 Creating a new cluster

2. The Create New Server Cluster Wizard is shown in Figure 9-5. Click **Next**.



*Figure 9-5 Create New Server Cluster Wizard*

3. In the Cluster Name and Domain window, select the domain where the cluster will be created and choose a new name for the cluster. This name will be registered in the domain and linked to a highly available IP address. See Figure 9-6.

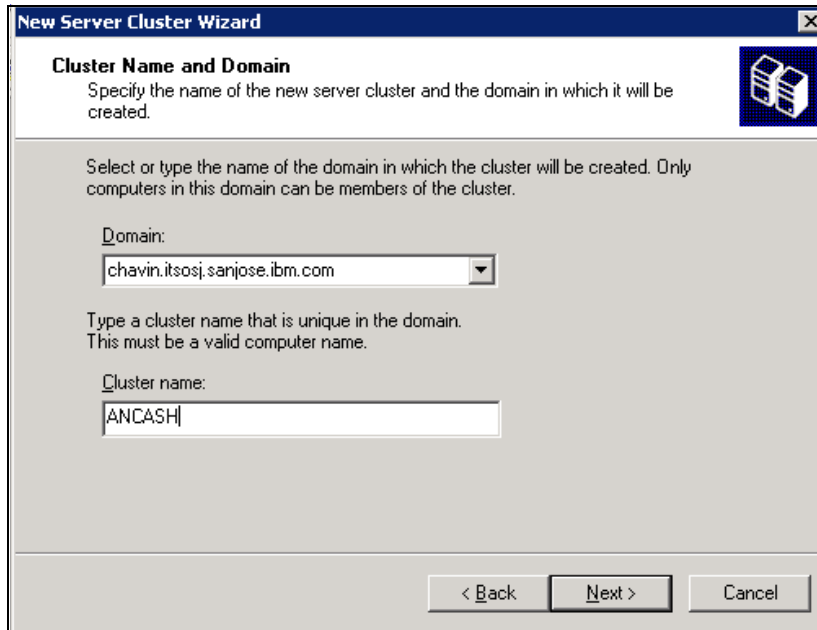


Figure 9-6 New Server Cluster Wizard

4. Select the Computer window shown, enter the computer name and click **Advance**.
5. In the Advanced Configuration Options window (Figure 9-7), you can select either configuration. In this example, we chose **Advanced (minimum) configuration** to configure every component.

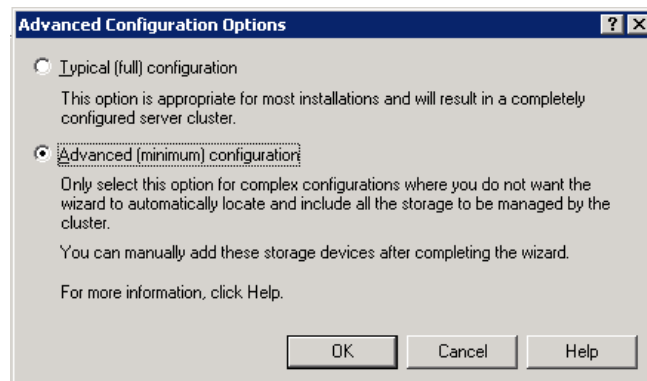


Figure 9-7 Advanced Configuration Options

6. If you have more than one shared disk, click **Quorum** in the Proposed Cluster Configuration window to select the disk drive as the Quorum device, or click **Next**. See Figure 9-8.

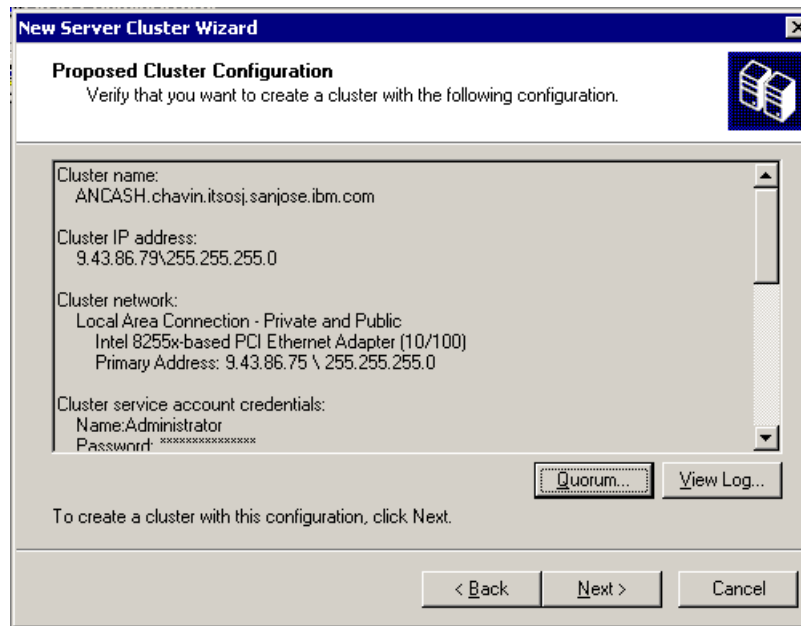


Figure 9-8 Proposed Cluster Configuration

7. If you click **Quorum** in the Proposed Cluster Configuration window to set up the Quorum, a Cluster Configuration Quorum window pops up. See Figure 9-9. In this window, choose one to be the Quorum device for this cluster. In our example, we chose Q drive.



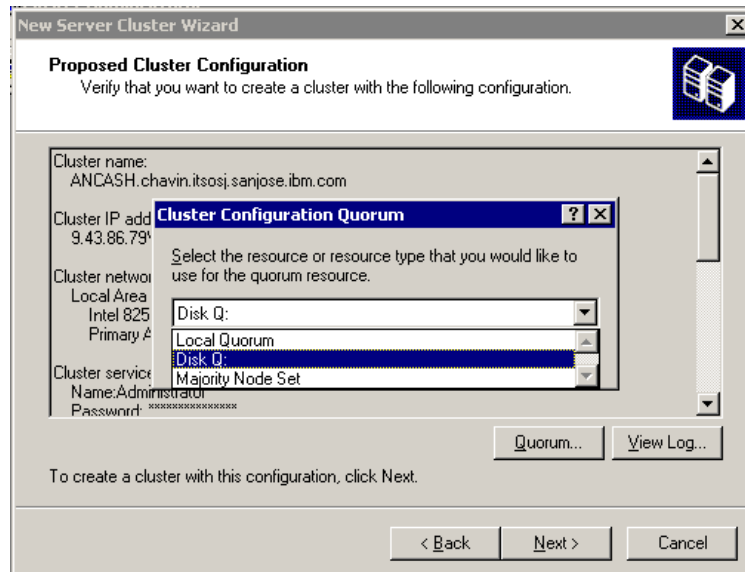


Figure 9-9 Choose the Quorum device

8. Now you have set up a cluster with one node. See Figure 9-10.

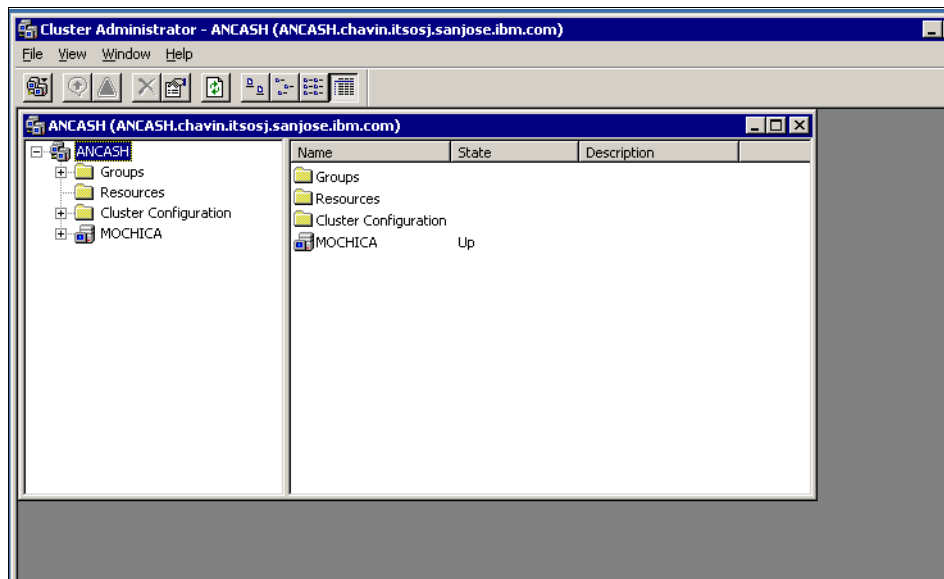


Figure 9-10 Cluster is set up with one node

9. From Windows Explorer, expand the Quorum drive. If the cluster is set up successfully, the MSCS directory is created and the directory contains the quorum log file quolog.log and a temporary file. See Figure 9-11.

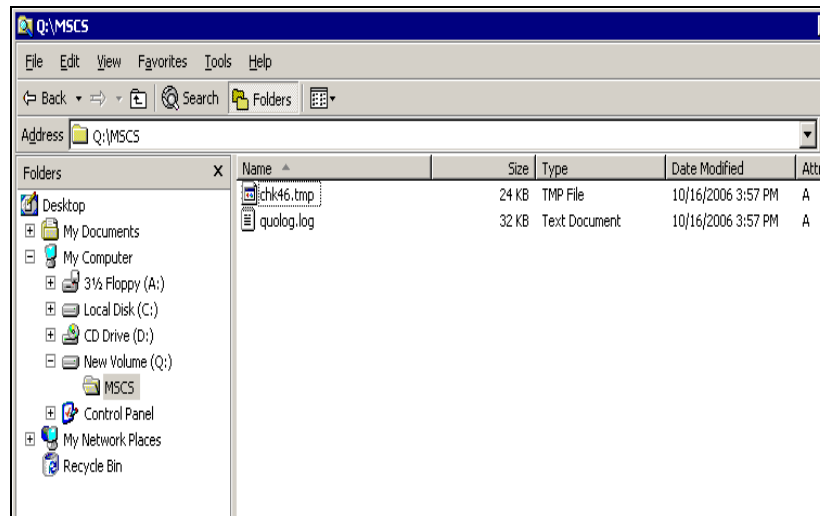


Figure 9-11 MSCS directory

The next step is to add the second node using the following steps:

1. To add a new node, connect to the cluster from **Start** → **Administrative Tools** → **Cluster Administrator**. In the Open Connection to Cluster window, enter the cluster name just created and click **OK**.
2. In the Cluster Administrator for the cluster window, select **File** → **New** → **Node**. The Add Node Wizard welcome window is shown. Click **Next**.

3. In the Select Computer window, enter the machine name to be added to the cluster in the Computer Name field and click **Add**. The machine name should be shown in the Selected computer area. See Figure 9-12.

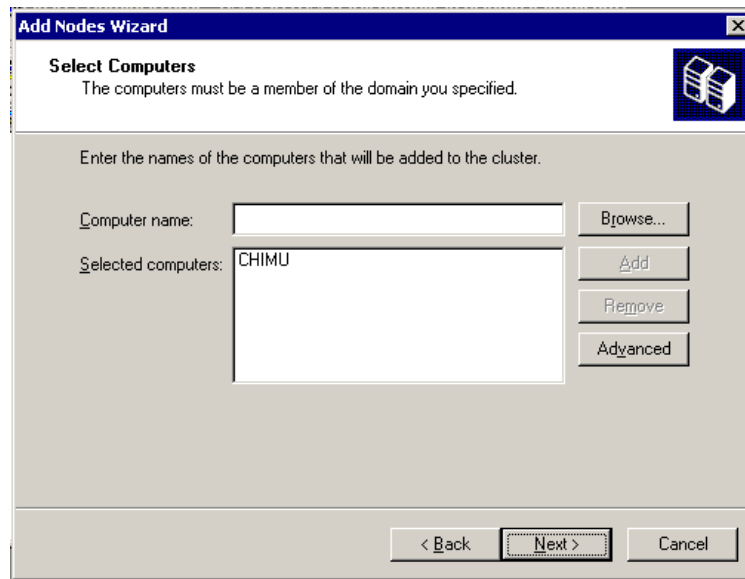


Figure 9-12 Enter new node name

4. The Add Node Wizard analyzes if it is possible to add the node by checking the available resources. In this case, there is a warning message as the machine we used had only one network adapter and two is the recommended amount. See Figure 9-13. Click **Next**.

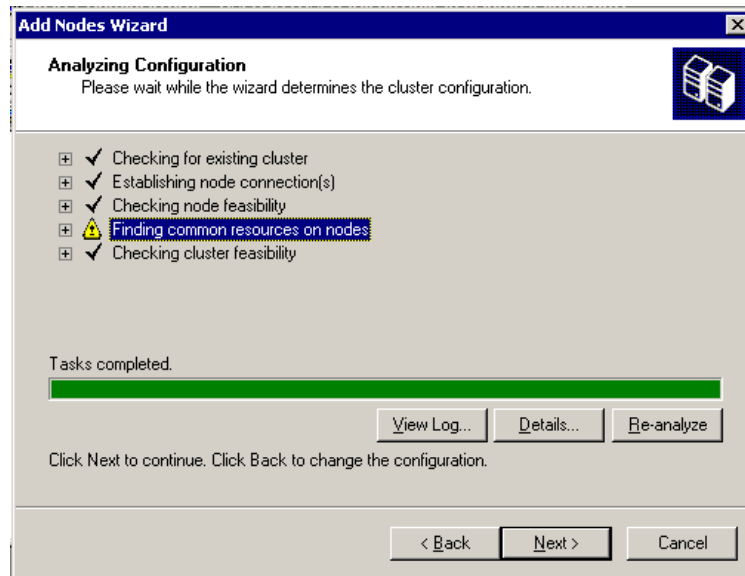
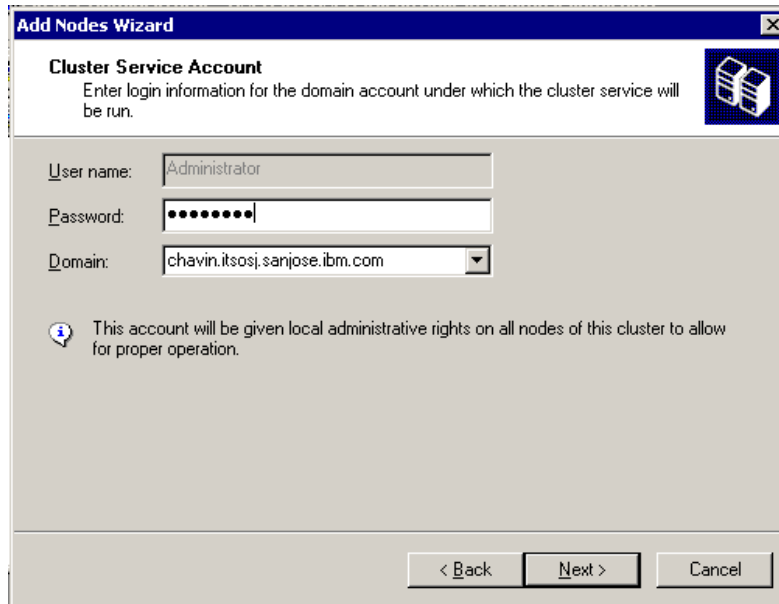


Figure 9-13 Adding a new node

5. The Administrator user and password on the domain are required to complete the operation (Figure 9-14). After this step, the new node should be added to the cluster.



The screenshot shows a Windows dialog box titled "Add Nodes Wizard" with a close button in the top right corner. The main heading is "Cluster Service Account" with a sub-instruction: "Enter login information for the domain account under which the cluster service will be run." To the right of the text is an icon of three server racks. Below the text are three input fields: "User name:" containing "Administrator", "Password:" with masked characters (dots), and "Domain:" with a dropdown menu showing "chavin.itsoj.sanjose.ibm.com". Below these fields is an information icon (i) and a note: "This account will be given local administrative rights on all nodes of this cluster to allow for proper operation." At the bottom right are three buttons: "< Back", "Next >", and "Cancel".

Figure 9-14 Cluster Service Account

Once the setup steps are completed in both nodes, verify that the files `%windir%\cluster\CLUSDB` and `%windir%\cluster\CLUSDB.log` exist in both machines. These are the database and the log file for the cluster respectively.

## 9.4 Installing DB2 ESE

DB2 should be installed on all the nodes in the cluster. For the DB2 installation procedure, refer to *Quick Beginnings for DB2 Servers*, GC10-4246.

You can create a DB2 instance along with a DB2 installation. We will defer instance creation to later on so we can specify all the options such as type, ports, and so on, as we want. In the Configure DB2 instances window, click **Next** to skip the DB2 instance creating step. See Figure 9-15.

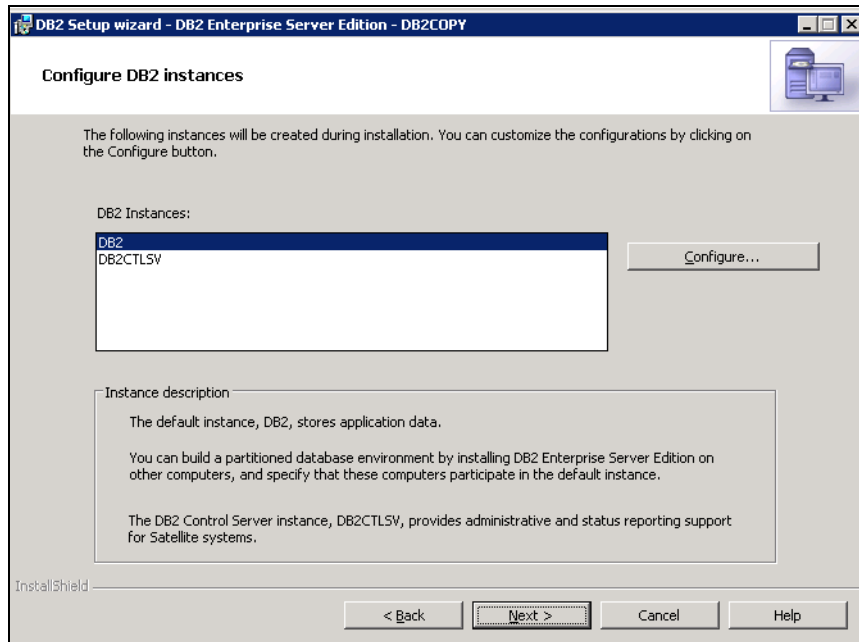


Figure 9-15 Skip the DB2 instance creating step

To check the DB2 installation quickly, verify if the following DB2 files exist:

- ▶ DB2 ESE version 8.2: The file %DB2PATH%\db2wolf.dll exists.
- ▶ DB2 ESE version 9.1: The %windir%\MSCS\db2server.dll exists.

## 9.5 Adding resources to a cluster for DB2

Initially, in the cluster, you will have a single group called Cluster Group that contains the following resources:

- ▶ Cluster IP address
- ▶ Cluster name
- ▶ Quorum drive

In our example, the highly available cluster IP address is set to 9.43.86.169 and the quorum device is Disk Q.

We have to add the shared disks as a resource for the cluster so the disks can be used to store the instance profile (in our example, we use Disk R:).

Use the Cluster Server adding resource to cluster steps to add the shared disk as follows:

1. In the Cluster Administrator utility, go to **File** → **New** → **Resource**.
2. In the New Resource window (Figure 9-16), enter:
  - a. Name: Disk R
  - b. Resource Type: Select **Physical Disk**. Then click **Next**.

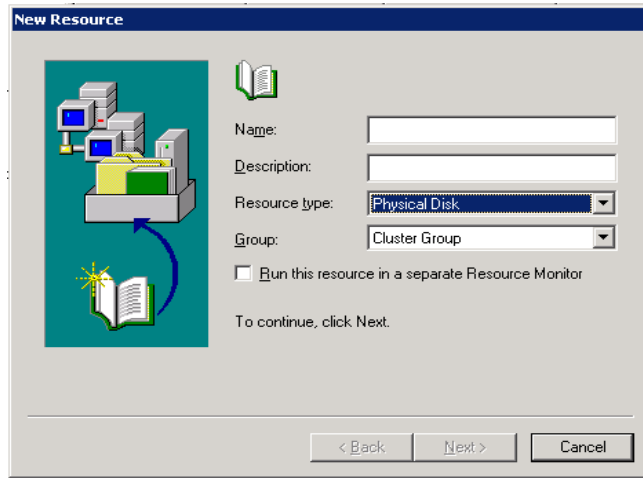


Figure 9-16 Add a new resource

3. The Possible Owners dialog is shown in Figure 9-17. By default, all nodes in the group will be selected. The order is not important, as the owner of the resource will be the group owner. Click **Next**.

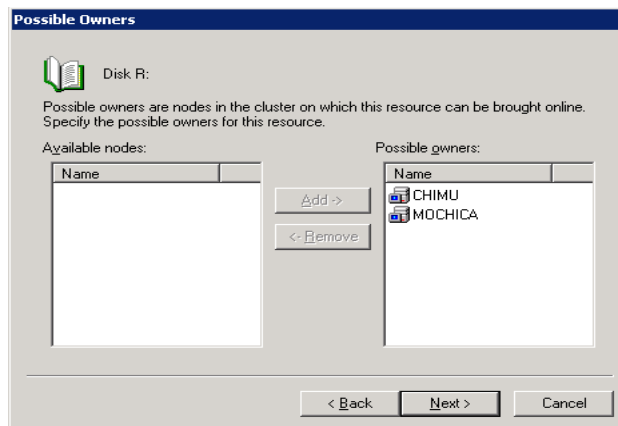


Figure 9-17 Possible Owners

4. You will be presented with the Resource Dependency dialog. If the resource that you are adding requires any other resource in order to work properly, you must add that resource in this dialog. For example, if the resource that you are adding is a DB2 instance, you will have to add the disks on which the DB2 instance is created to the dependency list. This way, you will avoid the possibility that DB2 is started before the disks are ready.

In this example, we are adding a physical disk that does not depend on any other resource. Therefore, the dependency list is empty (Figure 9-18).

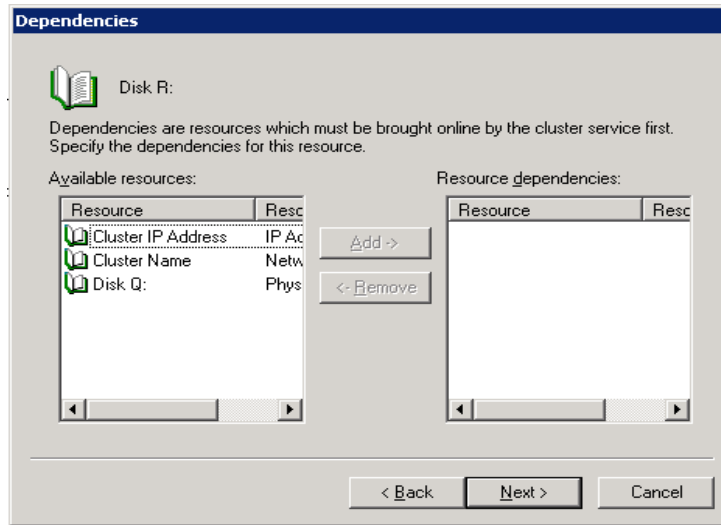


Figure 9-18 Resource Dependency

5. The last step to add a resource is to specify the resource-specific parameters. For example, if you are creating a highly available IP address, in the last step you will be asked for the specific IP address you want to use and the netmask.



Since we are adding a physical disk, the last step is a drop-down list box with all the disks available in the system (Figure 9-19). In this case, it is Physical Disk R.

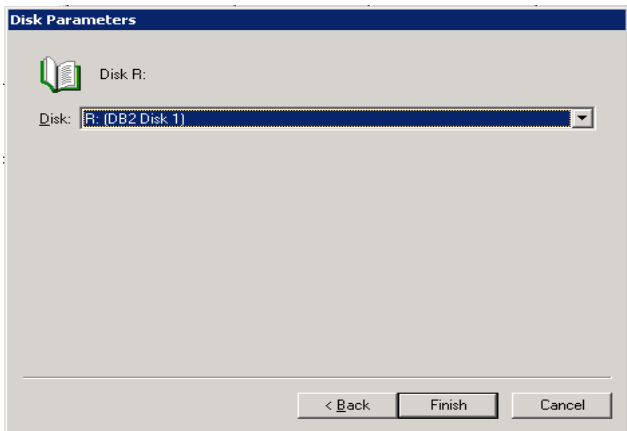


Figure 9-19 Resource Parameters

From the Cluster Administrator, we can see that the physical disk has been added successfully as shown in Figure 9-20.

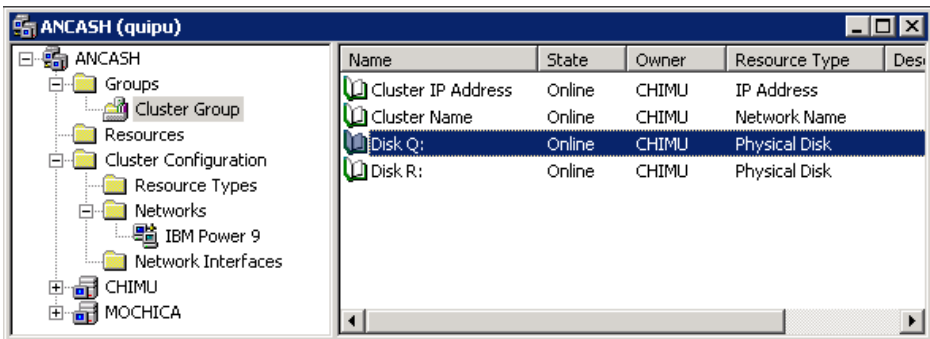


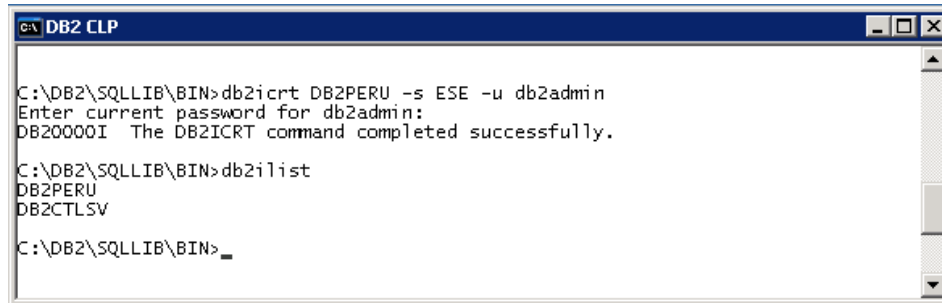
Figure 9-20 After adding a physical disk

## 9.6 Creating a DB2 instance

We create a single partition ESE instance using the db2icrt utility in the node to be used as the initial resource owner in the cluster. In our example, we created a DB2 instance in Mochica. The instance name is DB2PERU and the owner of the instance is the Domain user db2admin.

Figure 9-21 shows the command used to create the instance and the result of the command. Note that **db2icrt** will request the password of the domain user ID specified in the -u option.

The second command **db2ilist** in the same screen lists the instances in the local machine. Use the -w flag in the **db2icrt** command to set the type of instance to a Workgroup server or Enterprise server edition. Additionally, you can use the -p option to assign a port to the instance if necessary.



```
C:\DB2\SQLLIB\BIN>db2icrt DB2PERU -s ESE -u db2admin
Enter current password for db2admin:
DB20000I The DB2ICRT command completed successfully.

C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU
DB2CTLSV

C:\DB2\SQLLIB\BIN>
```

Figure 9-21 DB2 instance creation

If you open the Windows Services (**Start** → **Administrative Tools** → **Services**), you should see a new created service that represents the instance.

In the case of an ESE instance, the name of the service has the suffix **<INSTANCE-NAME>-<NODE NUMBER>**, for example, **DB2-DB2PERU-0**.

In the case of an WSE instance, the name of the service has suffix **<INSTANCE-NAME>**. If we created a WSE instance, the service will be like **DB2-DB2PERU**.

This name is very important, as it will be the Windows Registry Key Name of the service. This means that the following entries must exist in the Windows registry:

- ▶ ESE instance:  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DB2PERU-0
- ▶ WSE instance:  
HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\DB2PERU

The cluster software looks to this entry to start a highly available DB2 instance. This is usually the most common problem in setting up DB2 with Windows Cluster. If you name the DB2 resource with a name that is different from this entry, the resource will *not* find the DB2 server and will not start. Renaming the resource will not solve the problem and you will have to recreate it.

In the following sections, we discuss two different methods to make the DB2 instance highly available with Windows Cluster: the manual method and the automatic method using db2mscs.

Note that you cannot use **db2icrt** with option **-c <CLUSTERNAME>** to create a DB2 instance directly under the specified cluster. To create an instance in the cluster, you must create a DB instance and then migrate the instance to the cluster with the procedures discussed here. If you try to create the instance with the **-c** option, you will get a syntax error and the instance will not be created.

## 9.7 Manually configuring a DB2 instance in Windows Cluster

Although there is a semi-automatic utility db2mscs that does most of the work for you, it is important to understand what steps are involved in making a DB2 instance highly available in a Windows Cluster environment, so that you will be able to diagnose any problem that occurs.

The following procedure can be used to configure a DB2 instance in a Windows cluster to provide high availability:

- ▶ Add a DB2 resource type to the cluster.
- ▶ Create a Cluster group to group all the resources for the instance.
- ▶ Create or move the supporting resources to the DB2 group:
  - A highly available IP address is mandatory.
  - A highly available shared disk to store an instance profile is mandatory.
  - If this is a multi-partition database, the instance profile directory must be shared and made highly available.
- ▶ Move the DB2 instance to the cluster.
- ▶ Add the instance to the remaining nodes.

### 9.7.1 Adding the DB2 resource type

Once DB2 is installed and the instance is created, you have to create a DB2 resource type in Cluster Server. A resource type is basically a dynamic load library that is registered in Windows Cluster to manage a specific type or types of resource. To allow Windows Cluster to manage the DB2 instance, the DB2 instance must be a resource in the cluster.

In the Cluster Administrator, under Cluster Configuration, the Resource Type folder contains the current resource types that can be managed by the cluster server and the DLL that manages them. See Figure 9-22.

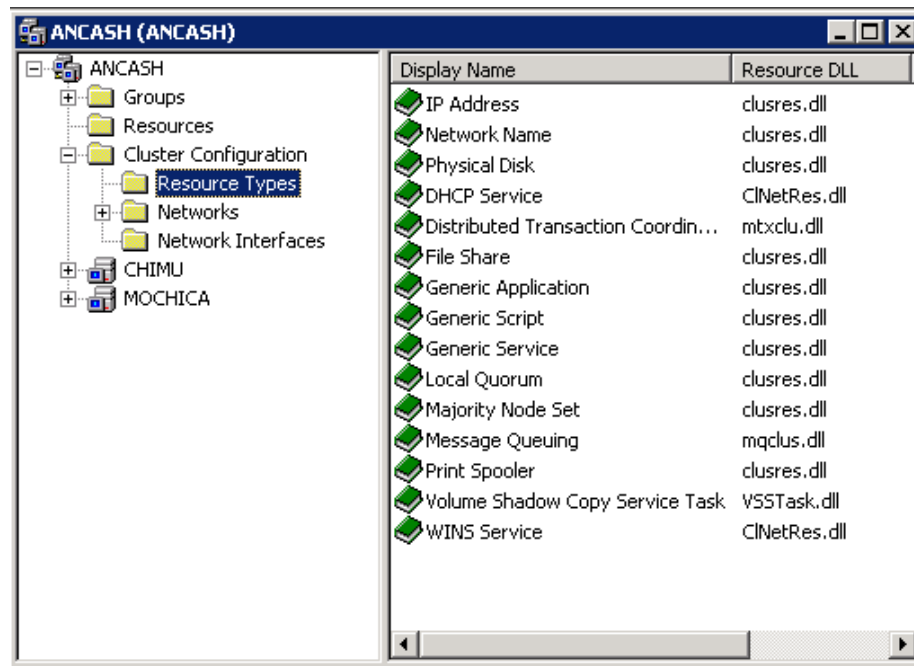


Figure 9-22 Initial resource list in Cluster Administrator

The DB2 resource type can be added to this list using the *db2wolfi* utility. This utility accepts one parameter:

**db2wolfi u | i**

Where:

- ▶ **u**: Unregister the DB2 resource type
- ▶ **i**: Installs the resource type

Figure 9-23 shows the registration of the DB2 resource type. If you try to register a resource that was already registered, you will get the error code 183.



Figure 9-23 Adding DB2 resource type

After db2wofli is run successfully, the new resource type DB2 is added. See Figure 9-24.

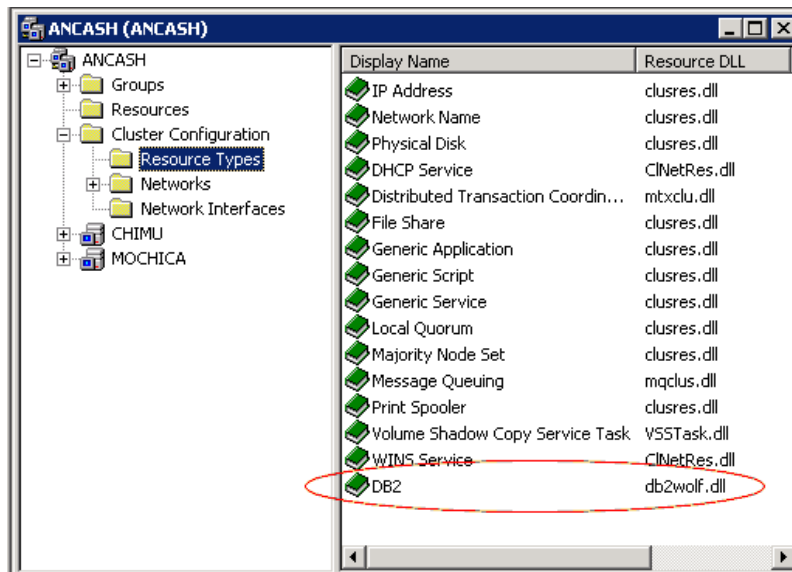


Figure 9-24 DB2 resource type added

Once DB2 is added as one of the resource types, Windows Cluster Server can manage a DB2 instance to provide highly available service.

You can see from Figure 9-24 that the DB2 resource type is managed by library db2wolf.dll. This is the case in version 8.2. In DB2 version 9.1, the resource is managed by library db2server.dll located in %WinDir%\MSCS. db2server.dll in turn invoke the appropriate db2wolf.dll in the DB2 instance directory.

## 9.7.2 Creating a DB2 group

In order to create a highly available DB2 resource, we recommend that you place this resource in a separate group that can be failed over, failed back, and made online/offline independently from the main cluster group.

The group can have any name. The recommended naming convention is to name it after the instance. In our example, the DB2 instance is DB2PERU. We create a group called DB2PERU-0 Group.

To create a group in the cluster, follow these steps:

1. From Cluster Administrator utility go to **File** → **Cluster** → **New** → **Group**.
2. The New Group wizard starts. Enter the group name in the Name field. We name this group DB2PERU-0 Group. See Figure 9-25.

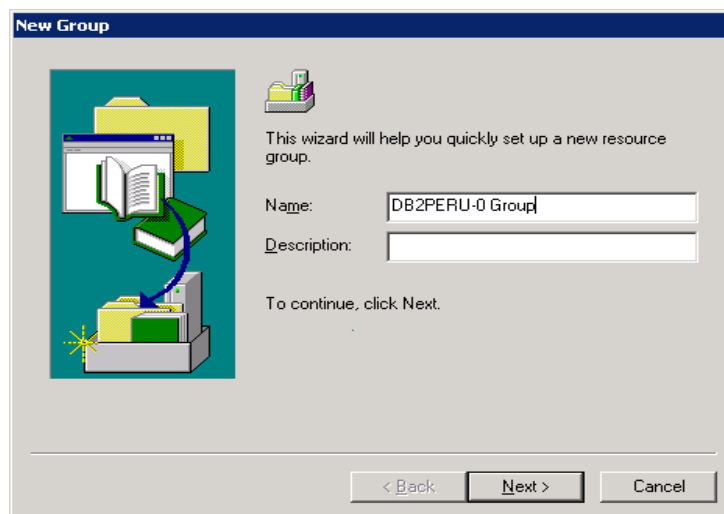


Figure 9-25 New Group wizard

3. In the Preferred Owners window (Figure 9-26), add the nodes that will be the owners of this resource.

**Note:** The node sequence makes a difference here. The node at the top of the list in the Preferred Owners list will be the initial owner of the group. All resources created inside a group are initially owned by the group owner.

The owner of the resource is the node that can use the resources. In the case of a shared disk, for example, even the disk is shared and both nodes have paths to access it. The cluster software makes it so only the current owner can read and write that device.

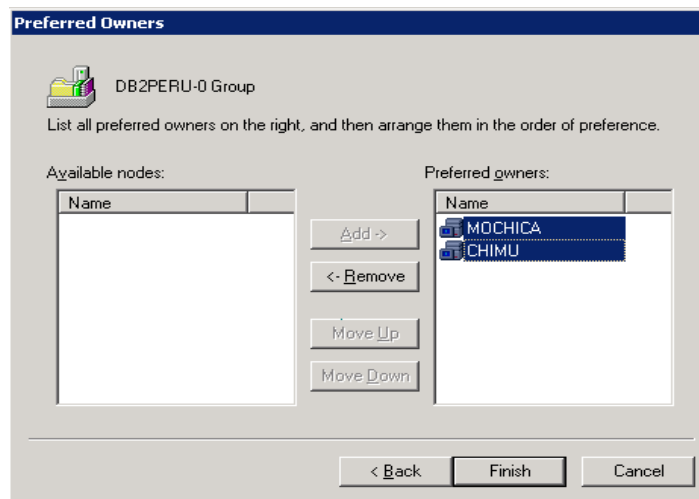


Figure 9-26 Owners

Once the group is created, you will notice that there is a red arrow next to the group name that indicates that the group is offline (not working). This is due to the fact that the group does not have any resource defined and therefore is not working.

You can now move the resource Disk R: created before into the new group. We will use this resource to store the instance profile. You can move the resource to a group by just dragging the resource from the Cluster Group folder and drop it into the new group. Respond **Yes** to confirm the action. Once the group contains a resource, the group will be up automatically. Figure 9-27 shows that DB2PERU-0 Group contains resource Disk R.

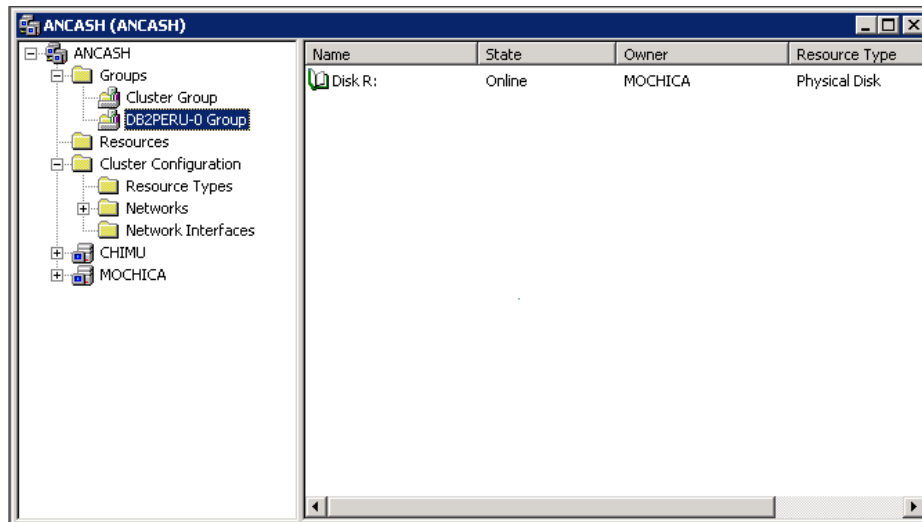


Figure 9-27 DB2 group created

### 9.7.3 Creating a highly available IP address for the DB2 resource

DB2 clients access the DB2 server via TCP/IP. The IP address, which the DB2 client used to communicate with the DB2 server, also has to be included in the Window Cluster resource group. In the event of a machine failure, the other node will take over the IP address, and clients will be able to attempt a reconnect.



To add an IP address to the cluster, follow the steps described in 9.5, “Adding resources to a cluster for DB2” on page 232. Enter the resource name and select **IP Address** as the Resource type as shown in Figure 9-28.

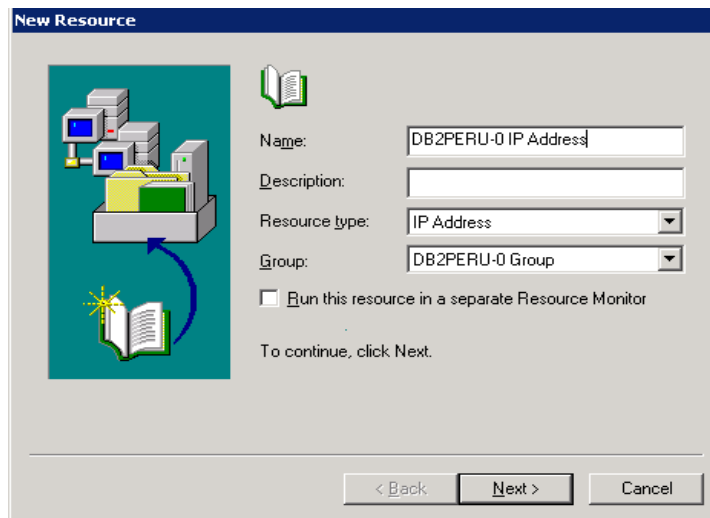


Figure 9-28 Creating IP Address resource

In the last step, you will be asked the IP address to use as well as the netmask. See Figure 9-29. After the creation is complete, you will have to bring the resource online by right-clicking the resource and selecting **Bring online**.

After you see that the IP address is online, try to ping it from another machine to ensure that it is working properly.

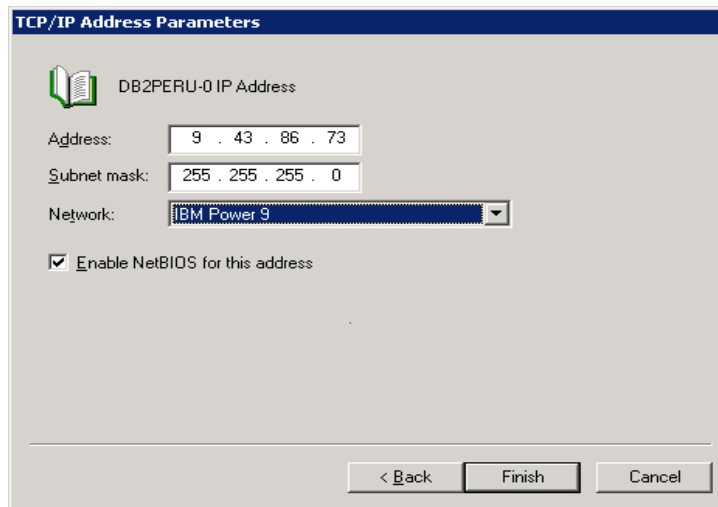


Figure 9-29 New IP Address resource created

## 9.7.4 Migrating the DB2 instance to the cluster environment

The instance, as it is now, has been created in only one node in the cluster. In the event of a failure, the other node would not have been able to find information about this instance. The same applies for any other resource in the cluster. Windows Cluster maintains information about the status, resources, resource types, nodes, and so on, of the cluster in a file located in %WinDir%\Cluster\CLUSTDB. This file exists in each node and is synchronized constantly by the Cluster Server software.

Additionally, the applications in the cluster have to access configuration information in any of the nodes. To avoid replicating potentially multiple registry keys in the registry, most of the contents of the CLUSTDB and all information related to a highly available resource must be located in the same Windows Registry key:

**HKEY\_LOCAL\_MACHINE\Cluster**

This key is also synchronized among the nodes by Windows so applications can use it in any of the nodes.

The process of migrating a DB2 instance to the cluster includes two steps:

1. Moving the DB2 instance profile directory to a highly available disk where it can be found by any node.

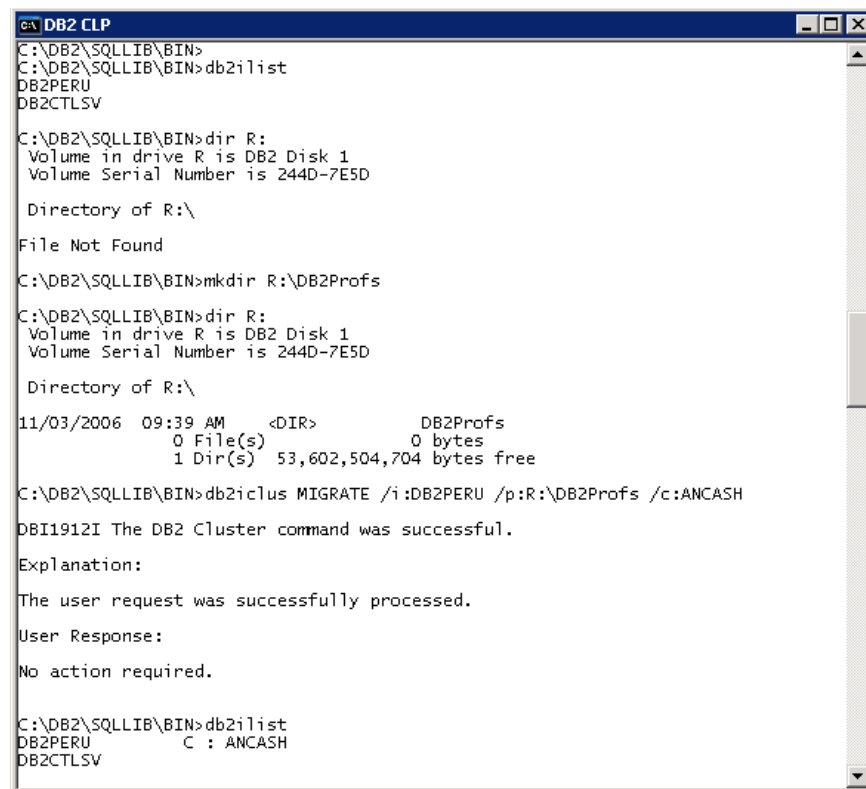
2. Creating a Windows register key in the cluster section of the Windows Registry.

These two tasks are accomplished using the *db2iclus* utility that must be run in the node where the DB2 instance was created.

In Figure 9-30, you can see a series of commands:

- ▶ **db2ilist** shows you that there are two instances in the system. None of them are available in the cluster.
- ▶ **mkdir R:\DB2Profs**: In the highly available disk Drive R, we create a directory called DB2Profs to store the instance profiles.
- ▶ **db2iclus MIGRATE /i:DB2PERU /p:r:\DB2Profs /c:ANCASH**: The **db2iclus** command moves the instance DB2INCA to the cluster ANCASH and places the DB2 instance profile directory in the directory R:\DB2Profs.

The last **db2ilist** command shows that the instance has been moved to the cluster as we required.



```
C:\DB2\SQLLIB\BIN>
C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU
DB2CTLSV

C:\DB2\SQLLIB\BIN>dir R:
Volume in drive R is DB2 Disk 1
Volume Serial Number is 244D-7E5D

Directory of R:\

File Not Found

C:\DB2\SQLLIB\BIN>mkdir R:\DB2Profs

C:\DB2\SQLLIB\BIN>dir R:
Volume in drive R is DB2 Disk 1
Volume Serial Number is 244D-7E5D

Directory of R:\

11/03/2006  09:39 AM    <DIR>          DB2Profs
               0 File(s)                0 bytes
               1 Dir(s)  53,602,504,704 bytes free

C:\DB2\SQLLIB\BIN>db2iclus MIGRATE /i:DB2PERU /p:r:\DB2Profs /c:ANCASH
DBI1912I The DB2 Cluster command was successful.

Explanation:
The user request was successfully processed.

User Response:
No action required.

C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU      C : ANCASH
DB2CTLSV
```

Figure 9-30 DB2 Instance migration

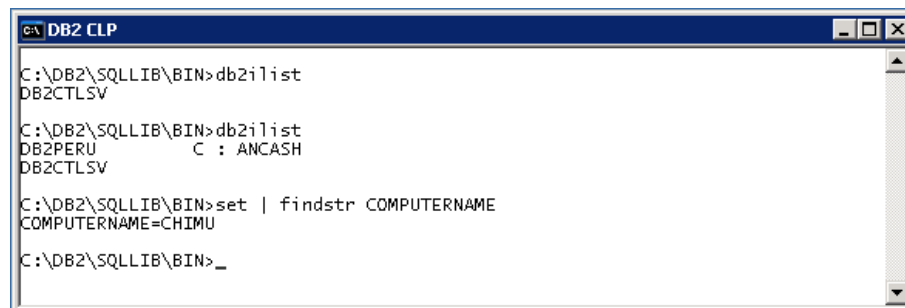
## 9.7.5 Adding a reference to the instance in the other nodes

The **db2iclus migrate** command places the DB2 instance profile directory in the shared drive and adds the instance to the shared part of the Windows registry. However, the instance was created in one node only (Mochica in this example). The instance creation process modifies several Windows registry keys as well as creating the service to be started in Windows. All these elements are missing from the other nodes in the cluster. If a failure occurs at this point, the cluster server will correctly failover to another node (Chimu) but Windows will not find the Windows Service for the DB2 instance DB2PERU. The failover process will be aborted.

Use the **db2iclus** command with the **ADD** clause to create references to the instance in the remaining nodes. In our example, the second node is Chimu. The command must be run from the node that has the instance to be referenced and is as follows:

```
db2iclus ADD /i:DB2PERU /m:CHIMU /c:ANCASH /u:db2admin
```

Figure 9-31 shows that the **db2list** command is executed twice in machine Chimu. The second execution is after running the **db2iclus ADD** command showing that the instance is now existing in this node.



```
C:\DB2\SQLLIB\BIN>db2list
DB2CTLSV

C:\DB2\SQLLIB\BIN>db2list
DB2PERU          C : ANCASH
DB2CTLSV

C:\DB2\SQLLIB\BIN>set | findstr COMPUTERNAME
COMPUTERNAME=CHIMU

C:\DB2\SQLLIB\BIN>_
```

Figure 9-31 Adding a DB2 instance to the second node

## 9.7.6 Creating the highly available DB2 instance resource

Migrating the instance to a cluster environment does not add this DB2 instance as a resource for the cluster. That is, up to this point, the cluster does not know that there is a highly available DB2 instance to be managed. In order to accomplish this, you must create a resource of type DB2 using Cluster Administrator utility.

Use the New Resource wizard in the Cluster Administrator to add the DB2 instance as a cluster resource:

1. From the Cluster Administrator, right-click the DB2 group created (DB2PERU-0 Group in this case) and then **File** → **New** → **Resource**.
2. The New Resource wizard shows up.

The resource name must be *exactly* the same as the last part of the Windows service name of the DB2 instance. In our example, the Windows service name of our instance is DB2-DB2PERU-0, so the resource name must be DB2PERU-0. Select **DB2** as the Resource Type and click **Next**. See Figure 9-32.

**Note:** If the resource name created is incorrect, the Cluster Server will not be able to find the service to start or stop DB2. The resource will be offline.

To change the resource name, you have to drop and recreate the resource with the correct name. Renaming the resource will not solve the problem.

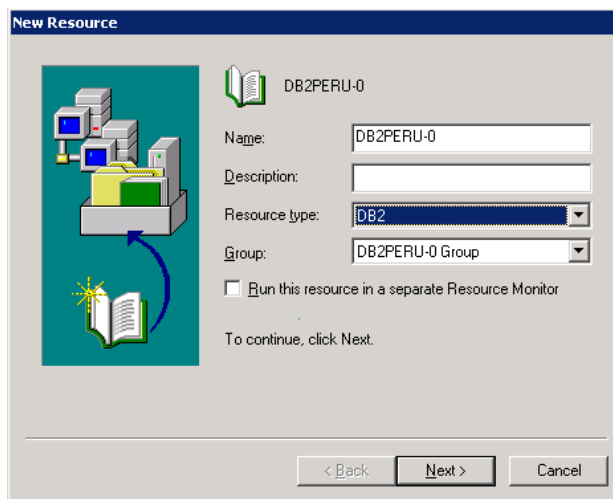


Figure 9-32 Creating the DB2 instance resource

3. In the Possible Owners window, accept the default of all nodes in the cluster as the owners. Click **Next**.
4. The Dependencies window is shown. The instance must be dependent in the highly available IP Address and disks created, as the instance cannot start without these two resources. They are DB2PERU-0 IP Address and Disk R:.

See Figure 9-33. Click **Next** to proceed. In the next dialog, click **Finish** to create the resource.

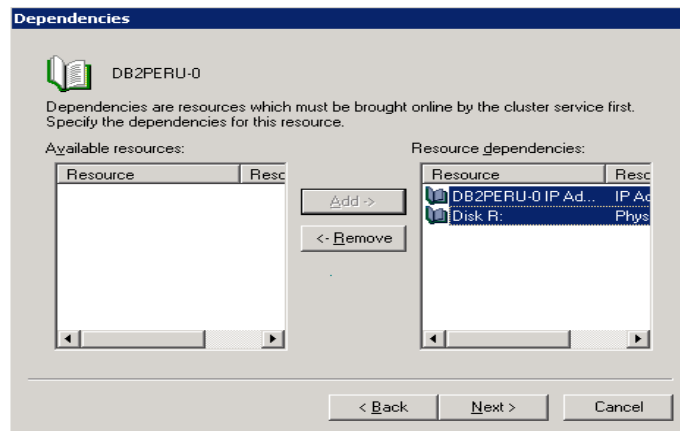


Figure 9-33 Dependency list for the DB2 instance resource

5. In the Cluster Administrator main window, right-click on the newly created resource and click **Bring online**.

Once the DB2 instance is added to the cluster, you should start/stop the DB2 instance using only the Cluster Administrator interface. If you use any other means such as the DB2 Command Line Processor, you might have the following problems:

- ▶ If you stop the instance from the command line processor, the Cluster Server will detect this as a failure and will try to bring the instance online immediately.
- ▶ If the DB2 cluster service is offline and you bring the instance online from the command line processor, the cluster will not detect this and thus the instance will not be highly available.

## 9.8 Configuring a highly available DB2 instance using db2mscs

The *db2mscs* utility automates most of the steps of creating a highly available DB2 instance in the Windows Cluster environment. The prerequisite is that the shared disk where the instance profile will be moved must be made highly available before running the utility. All other steps such as creating the highly available IP address and network name are automated by the tool.

**db2mscs** uses a configuration file that defines the user and password to create the services, the name and value of the resources like IP Addresses and network names, and so on. **db2mscs** accepts three parameters:

**db2mscs -f:<config\_file> -u:<instance name> -d:<debug\_file>**

- ▶ **-f** option: This option specifies the configuration file. The default value of this parameter is db2mscs.cfg
- ▶ **-u** option: This option allows you to revert all changes in the specified instance and return it to its unclustered form.
- ▶ **-d** option: This option specifies the debug file name. The utility writes debugging information into the text file specified in this option.

The configuration file accepts parameters in the following format:

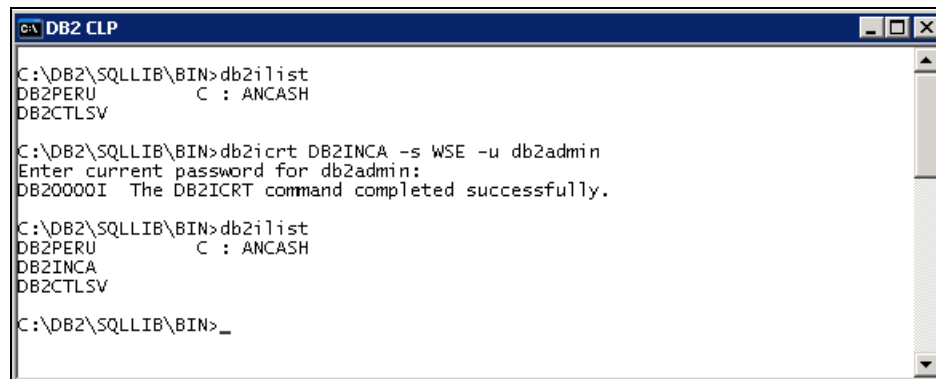
PARAMETER	VALUE
-----------	-------

DB2 provides some sample configuration files under the directory  
%DB2Dir%\sqllib\cfg:

- ▶ db2mscs.das, to cluster a DAS instance.
- ▶ db2mscs.ee, to cluster an EE instance.
- ▶ db2mscs.eee, to cluster an EEE instance.

The meaning of each parameter can be found in the DB2 manual, *Command Reference*, SC10-4226.

Since we have created an ESE instance in the example above, we use db2mscs to cluster a WSE instance. Figure 9-34 shows the commands to a DB2 WSE instance named DB2INCA.



```
C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU      C : ANCASH
DB2CTLSV

C:\DB2\SQLLIB\BIN>db2icrt DB2INCA -s WSE -u db2admin
Enter current password for db2admin:
DB20000I  The DB2ICRT command completed successfully.

C:\DB2\SQLLIB\BIN>db2ilist
DB2PERU      C : ANCASH
DB2INCA
DB2CTLSV

C:\DB2\SQLLIB\BIN>_
```

Figure 9-34 Instance creation

The sample configuration file that we use is shown in Example 9-1.

*Example 9-1 Windows Cluster configuration file*

---

```
DB2_LOGON_USERNAME=CHAVIN\db2admin
DB2_LOGON_PASSWORD=mypasswd123
CLUSTER_NAME=ANCASH
GROUP_NAME=DB2INCA Group
DB2_INSTANCE=DB2INCA
#DB2_NODE=0

IP_NAME=DB2INCA IP Address
IP_ADDRESS=9.43.86.131
IP_SUBNET=255.255.255.0
IP_NETWORK=IBM Power 9

NETNAME_NAME=DB2INCA Net
NETNAME_VALUE=DB2INCA Net
NETNAME_DEPENDENCY=DB2INCA IP Address

DISK_NAME=Disk T:
INSTPROF_PATH=T:\DB2Profs
```

---

Here are some points to notice in this example:

- ▶ The value of IP\_NAME is arbitrary. It is a good practice to use the instance name for the IP\_NAME.
- ▶ The value of NETNAME\_DEPENDENCY has to be the same as the one in IP\_NAME since the network name depends on the IP\_NAME.
- ▶ The value of DISK\_NAME must be a physical disk resource which has already existed in the cluster.

Before running db2mscs, you *must* set the environment variable DB2INSTANCE to be the same value as the db2mscs.cfg parameter DB2\_INSTANCE. If you fail to do this, the operation will fail because db2mscs tries to start the sentence specified by DB2INSTANCE. Typically you will receive an error message such as:

```
C:\DB2\SQLLIB\BIN>db2mscs -f:\temp\db2mscs.cfg
DB21516E DB2MSCS cannot bring resource "DB2CTLSV" online. Ensure that
the properties of the resource are set correctly.
```

Changing the instance name will solve the problem, as we can see here:

```
C:\DB2\SQLLIB\BIN>set DB2INSTANCE=DB2INCA
C:\DB2\SQLLIB\BIN>db2mscs -f:\temp\db2mscs.cfg
```

```
DB21500I The DB2MSCS command completed successfully.
```



Once these steps are completed, you can see the newly created resource in the Cluster Administrator as shown in Figure 9-35.

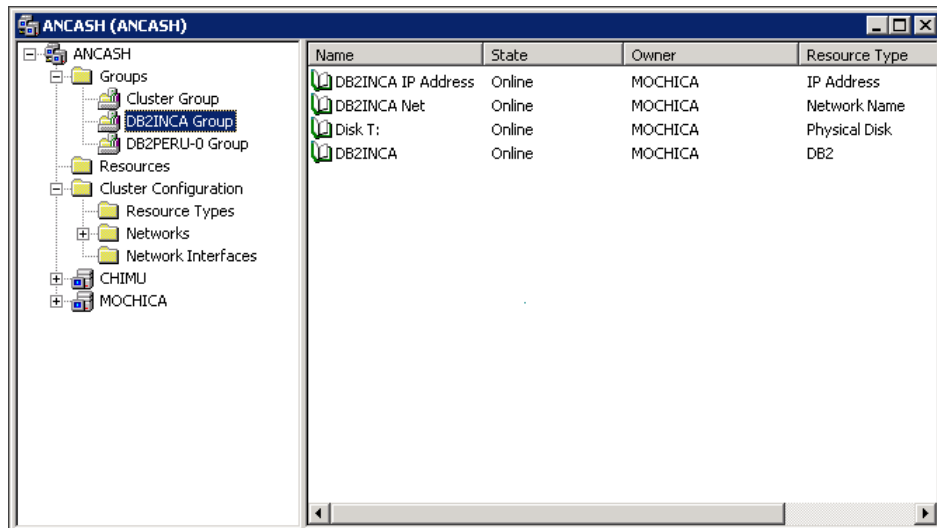


Figure 9-35 Final result of clustering a WSE instance





## HADR with clustering software

Clustering software such as High Availability Cluster Multi-Processing (HACMP) and Tivoli System Automation (TSA) are among system high availability options. Combining High Availability Disaster Recovery (HADR) with clustering software strengthens the high availability for the computing environment. In this chapter we discuss how to configure HADR with HACMP or TSA to enable automating HADR takeover.

We introduce the following topics:

- ▶ Why clustering software is needed
- ▶ Automating HADR takeover with HACMP
- ▶ Automating HADR takeover with TSA

## 10.1 Overview: Why clustering software is needed

The present version of HADR does not monitor the environment of the primary database server for outages such as network down. As illustrated in Figure 10-1, in the Remote Catchup Pending state, the standby database keeps waiting for log records to be transferred even though the primary database is no longer active. It is necessary to monitor the HADR pair and manually issue appropriate takeover commands in the event of a primary database server failure. This is where clustering software is required to automate HADR takeover.

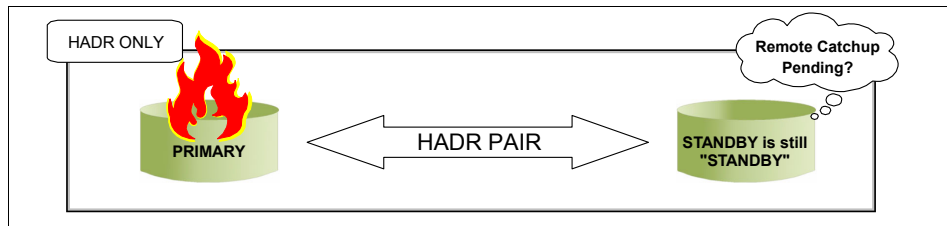


Figure 10-1 Takeover HADR will not happen automatically

### What is clustering software?

Clustering software automates failover of resources such as processes, applications, storages, and IP addresses. Clustering software monitors the health of the network, hardware and software processes, detects and communicates any fault, and automatically fails the service and associated resources over to a healthy host in the cluster.

In a cluster environment, the terms *failover*, *fallover*, and *takeover* are often used interchangeably, referring to the action whereby system activities are moved from the failed system to the healthy one. Specifically, failover refers to the activity of the broken node/server handing over responsibility to a backup or standby node/server. Fallover is similar, but a more general term for moving resources, including planned move operations for the purpose of maintenance. Takeover refers to the activity of the backup node/server making resources active after the original primary server breaks/fails.

There are several clustering software you can choose from depending on the platforms where the databases are running. The following lists a few of them:

- Tivoli System Automation for Linux

Tivoli System Automation is a solution that is designed to provide a multi-tiered architecture of clustering in a heterogeneous environment. This may either be through the TSA base component on various platforms, or through existing base-level cluster solutions from other vendors, coupled with the TSA end-to-end component, which interfaces with all the base-level

clustering software (with adapters for non-TSA products). This enables central control of all clusters, and facilitates the grouping of interdependent base-level clusters into applications. TSA provides a consistent, single monitoring and control interface for these disparate servers and clusters within a mission-critical business environment, and can truly be considered an integrated and automated highly available solution.

For detailed information about Tivoli System Automation, see the white paper entitled “Highly Available DB2 Universal Database - using Tivoli System Automation for Linux”, which is available on the Web site, “DB2 Database for Linux, UNIX®, and Windows and DB2 Connect - Online Support”:

<http://www.ibm.com/software/data/pubs/papers/>

- ▶ IBM High Availability Cluster Multi-Processing for AIX

HACMP V5 offers robust high availability and disaster recovery for IBM customers with mission-critical applications. HACMP provides base services for cluster node membership, system management, configuration integrity and control, and failover and recovery for applications. HACMP clusters with both non-concurrent and concurrent access can contain up to 32 nodes. A node is an AIX 5L operating system image, and it may be a System p, an RS/6000® server, or a logical partition (LPAR) of an applicable System p.

For detailed information about HACMP, see the following Web site:

<http://www-03.ibm.com/systems/p/software/hacmp.html>

- ▶ Microsoft Cluster Server for Windows operating systems

Microsoft Cluster Server (MSCS) was introduced in Windows 2003. The new replacement for MSCS has improved capabilities: It allows you to connect up to eight nodes in a cluster; and, additional technology to make possible geographically remote clusters has been added.

For information about MSCS, see the following white paper “Implementing IBM DB2 Universal Database V8.1 Enterprise Server Edition with Microsoft Cluster Server” on “DB2 Database for Linux, UNIX, and Windows Support” Web site:

<http://www.ibm.com/software/data/pubs/papers/>

- ▶ LifeKeeper for Linux/Windows

LifeKeeper for Linux/Windows is a clustering software provided by SteelEye Technology, Inc. LifeKeeper provides features and functions to monitor system and application health, maintaining client connectivity and providing uninterrupted data access. By maintaining the system uptime, LifeKeeper ensures the high availability for Linux/Windows applications.

More details of this product are available at:

<http://www.steeleye.com/products/>

The article, *DB2 UDB and SteelEye LifeKeeper for Linux - A High Availability Database*, introduces a sample configuration of integrating DB2 into LifeKeeper cluster environment. This article is available at the following Web site:

<ftp://ftp.software.ibm.com/software/data/pubs/papers/lk.pdf>

► **ClusterPerfect**

DNCWARE ClusterPerfect is the clustering software of Toshiba Solutions Corporation. This product provides a way of keeping computing resources available to users in the event of a hardware or software failure. By connecting multiple servers, other servers in the cluster take over the workload when a server goes down.

### **How HADR works in an environment with clustering software**

When a primary database server outage occurs, clustering software detects the failure and fails the resources from the primary over to the standby node. You can configure the failover scripts to issue HADR takeover on the standby database in connection with the resource failover. Figure 10-2 shows a typical failover flow in the HADR cluster automated by the clustering software.

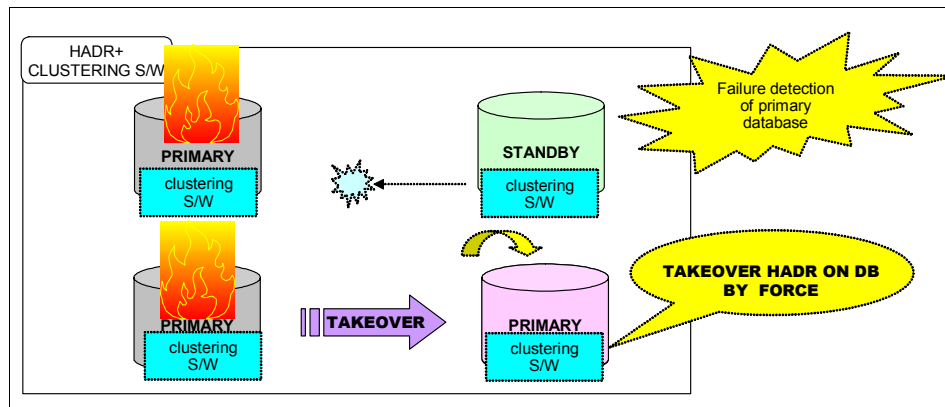


Figure 10-2 Why clustering software is needed to automate HADR takeover

In the cluster environment, the typical failover flow is as follows:

1. Under normal operations, users connect to the primary server and execute transactions against the database on this server. HADR continuously transfers the changes from the DB2 log from the primary to the standby server. In the standby, the logs are continuously replayed, thereby replicating the primary database. The standby database is in rollforward pending and no user can access this database.

2. In the event of an outage on the primary server, clustering software detects the fault and starts to failover the defined resources like IP address and executes takeover scripts. Clustering software can be configured to detect not only the server outage but also an instance crash by monitoring instance process.
3. The takeover script(s) issues an HADR takeover command against the standby database to change its role to primary. The old standby server can now serve users as the primary server.
4. While HADR takes care of synchronization of data between a primary and standby pair, there is still a need for applications to dynamically access the new primary server after an outage. This is achieved by the DB2 Automatic Client Reroute feature. Whenever the client fails to connect to the original server, the DB2 client will attempt to reroute the connection to the alternate server. If the reroute succeeds, the application can continue.
5. When the failed server comes back on line, it can then be started as the standby server, reversing roles to those prevailing prior to the outage.

### **What resources should be taken over**

The resources that must be taken over to automate HADR takeover with clustering software are as follows:

► **Storage devices:**

In a disk shared cluster, you must configure the shared storage device and create all database resources on it.

In an HADR environment, there is no need to share storage resources for failover, since the primary and standby databases are independent databases that reside on separate storage devices.

► **IP address:**

IP address takeover is optional. When you use Automatic Client Reroute, you do not always have to have a failover IP address because switching the IP address is handled on the client side. But be aware that for the clients or other servers which communicate with the database server and do not support Automatic Client Reroute, you must switch the definition of the IP address to the new primary database server.

► **Takeover scripts:**

Takeover scripts must be configured as a resource of the clustering software, and must be issued on the standby database by the clustering software at the time of takeover of resources after detecting an outage on the primary database.

## 10.2 Automating HADR takeover with HACMP

This section describes how to automate DB2 HADR takeover in an HACMP environment. The section also describes the failover test procedure for HADR controlled by the HACMP facility.

HACMP for AIX provides a highly available computing environment. HACMP facilitates the automatic switching of users, applications, and data from one system to another in the cluster after a hardware or software failure. The primary reason to create HACMP clusters is to provide a highly available environment for mission-critical applications. In an HACMP cluster, to ensure the availability of these applications, the applications are placed under HACMP control. HACMP ensures that the applications remain available even when a component in a cluster fails. To ensure availability in case of a component failure, the HACMP software moves the application along with the resources, to another node in the cluster. For more details regarding HACMP, see Chapter 8, “DB2 and HACMP” on page 183.

### 10.2.1 HACMP and HADR planning

Before you set up an HADR and HACMP environment, you must plan the cluster environment. The following items must be considered:

- ▶ Physical nodes
- ▶ Network
- ▶ HACMP configuration
- ▶ HADR configuration
- ▶ Scripts



## Lab environment

Figure 10-3 shows the configuration of HACMP and HADR in our Lab environment.

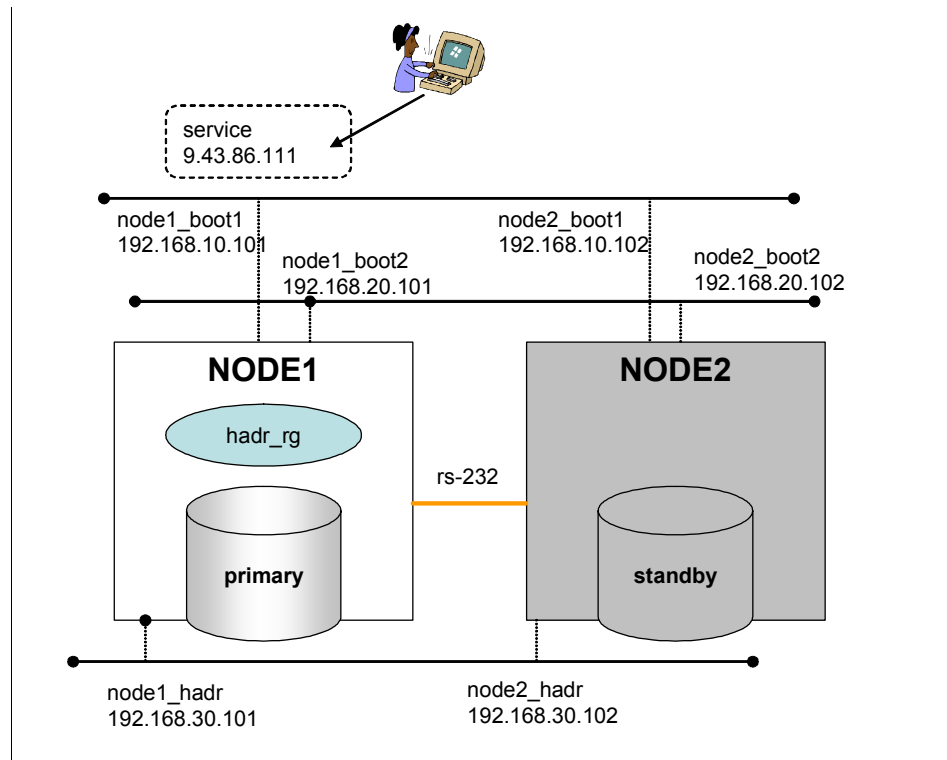


Figure 10-3 Lab environment

The planning for our Lab environment is as follows:

- ▶ Physical node configurations:
  - Two physical nodes named Node1 and Node2 are defined in the cluster.
  - Node1 is designated as the *service* node, which regularly provides service to clients. The HADR primary database runs on this node.
  - Node2 is the *standby* node on which the HADR standby database resides. The role of both these nodes changes in response to system failover or planned takeover issued by administrators.
- ▶ Network configurations:
  - Two ethernet network interfaces on each node are provided for the client's access, which are under the control of HACMP. The service address for the clients is added on one of these network interfaces.

- One ethernet network interface is dedicated to HADR communications. We recommend that you have a separate network for HADR if you want to avoid the interference of HADR log transfer.
- One serial (RS-232C) network is configured for HACMP keep-alive. Having a serial network (non TCP/IP network) is recommended in order to make HACMP failure detection more reliable.

**Note:** There is no need to have shared disks for HADR in an HACMP cluster, since the primary and standby databases are independent databases, which can reside in separate storage devices.

- ▶ HACMP configuration:
  - Resource group named *hadr\_rg* is configured, which includes a service address and an application server.
  - In our Lab, we configured a service address in the HACMP resource group. IP address takeover is optional if you use Automatic Client Reroute where clients can automatically switch the nodes they connect to.
- ▶ HADR configuration:
  - Each node has the DB2 instance named *hadrinst*. Instance names do not always have to be identical on both the nodes.
  - Each instance has the database named *SAMPLE*. The database name should be identical on both the nodes.
  - Configured dedicated network for HADR communication.

## Failover after primary node crash

Figure 10-4 shows the cluster behavior in the event of a primary (service) node crash:

1. HACMP detects the primary nodes outage and starts the failover process. The resource group *hadr\_rg* is acquired by the standby node (Node2).
2. The Service IP address related to the resource group is added to the standby node.
3. The Start script in the HACMP application server, which is related to the resource group, is issued on the standby node. The script includes the TAKEOVER HADR command to change the role of the standby database to primary.
4. Clients who address the service address succeed in connecting to the new primary database on the surviving node, which has the HACMP resource group now.

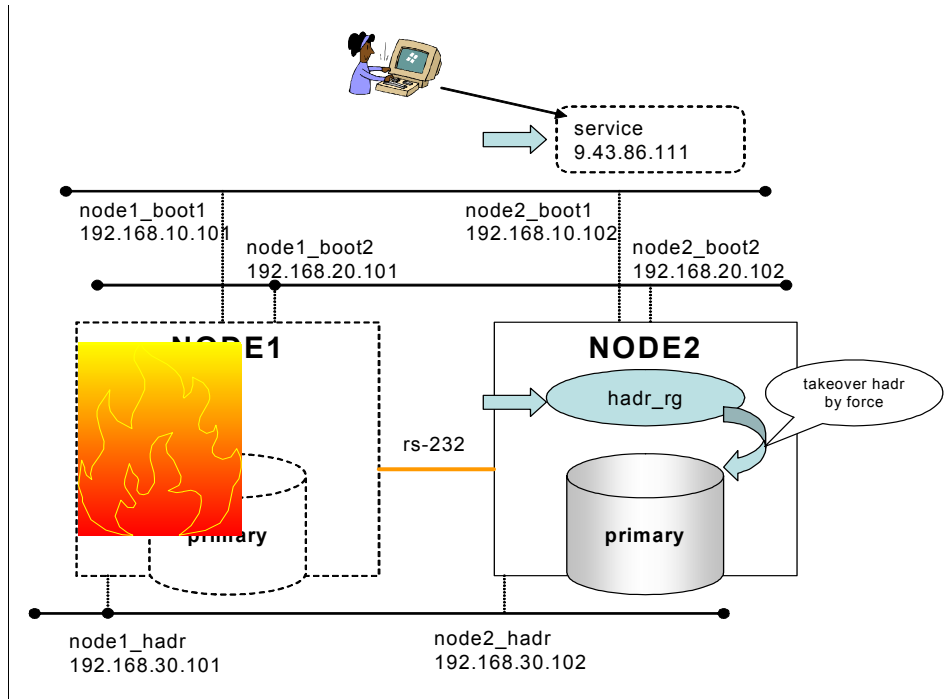


Figure 10-4 Automatic failover to standby node

## Reintegration of old primary node

Figure 10-5 shows the process required to reintegrate the old primary node into clusters. This process is as follows:

1. After Node1 is recovered from a hardware crash, you can start up instance and start HADR as the standby database on this node by executing the **start hadr** command with *as standby* option manually.
2. The standby database automatically catches up the log records which are processed only on the new primary database during the time the old primary Node is out of order.
3. After the standby database catches up all the log gaps, the HADR primary and standby again return to the Peer state. Reintegration of the old primary database hereby is complete.

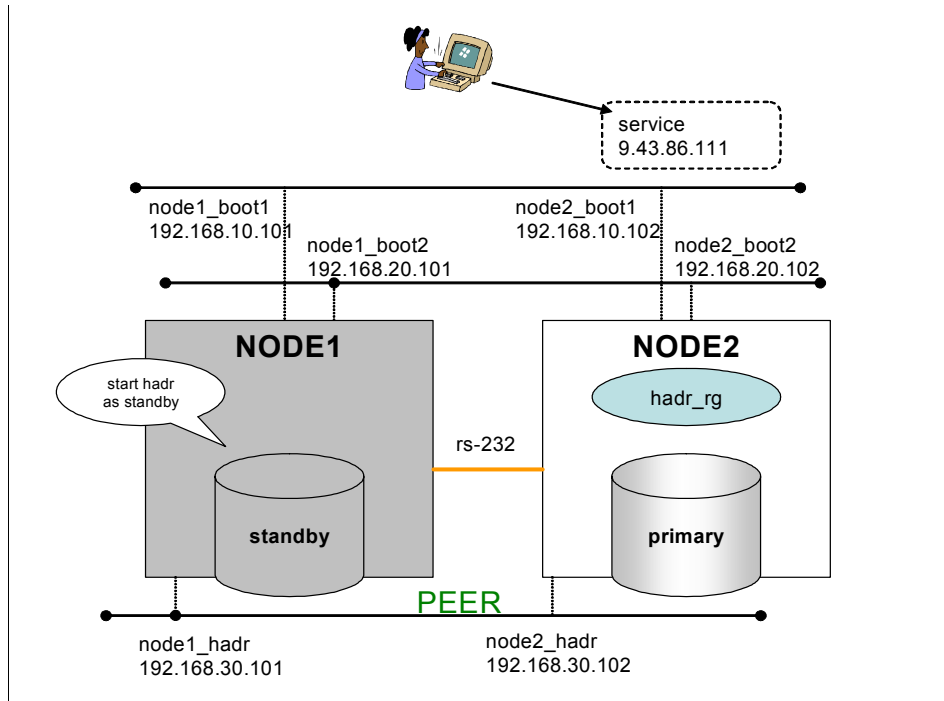


Figure 10-5 Reintegration of old primary database

## 10.2.2 Step-by-step configuration overview

You can configure automated HADR takeover environment with HACMP by completing the following operations:

1. Perform HADR setup.
2. Perform HACMP setup.
  - Here we test only the HACMP configuration and function with dummy scripts.
3. Prepare the scripts required to control HADR database from HACMP.
4. Perform a joint test of HADR and HACMP:
  - Normal start up of cluster
  - Planned takeover
  - Unplanned takeover
  - Normal stop of cluster

## 10.2.3 HADR setup

To configure the HADR database pair, complete the following steps:

1. For new systems, create a DB2 instance on both the nodes. We created `hadrinst`.
2. Check that proper entries are configured in `/etc/hosts` and `/etc/services` file.
3. For new systems, create a database on the primary node. We created `SAMPLE`.
4. Back up the primary database and restore the image on the standby node.
5. Configure HADR parameters properly on both the databases.
6. Start HADR on the standby database, and then start HADR on the primary database.
7. Check that both the databases can communicate with each other in the Peer state.

See Chapter 3, “HADR setup” on page 41 for more details on step-by-step configuration of HADR.

Figure 10-6 shows the entries for the services and hosts of the HADR configuration in our Lab environment.

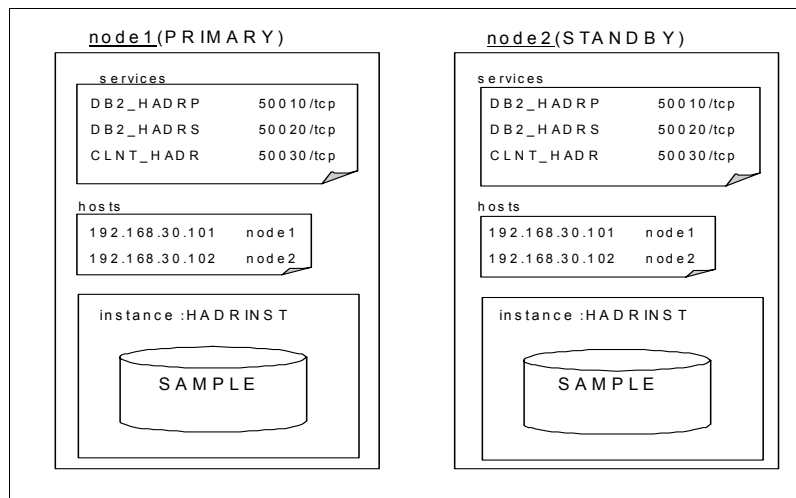


Figure 10-6 Instance, services, and hosts entries for HADR configuration

## 10.2.4 HACMP configuration

Here we describe the procedure to configure HACMP cluster for automatic failure detection of the primary node.

# Checking TCP/IP network connection

HACMP relies on a TCP/IP network connection to communicate between nodes. Before you configure HACMP, check that the network is configured properly. To check network configurations for the cluster complete the following steps:

- 1. Check whether the IP addresses are configured on network interfaces using the command:

```
#netstat -in | grep -v link
```

Example 10-1 shows the configuration of the IP addresses on both the nodes.

Example 10-1 netstat output

```
- node1 -
en0    192.168.10.101
en1    192.168.20.101

- node2 -
en0    192.168.10.102
en1    192.168.20.102
```

- 2. Check whether the /etc/hosts file has all the entries of IP addresses and that their labels are the same as used by HACMP:

```
#vi /etc/hosts
```

Example 10-2 shows an example of hosts file.

Example 10-2 IP entries in hosts file

9.43.86.111	service	## Service IP address
192.168.10.101	node1_boot1	## HACMP N/W interface on node1
192.168.20.101	node1_boot2	## HACMP N/W interface on node1
192.168.30.101	node1_hadr	## HADR N/W interface on node1
192.168.10.102	node2_boot1	## HACMP N/W interface on node2
192.168.20.102	node2_boot2	## HACMP N/W interface on node2
192.168.30.102	node2_hadr	## HADR N/W interface on node2

- 3. Verify that name resolution is working well by using the **host** command. If something is wrong, check and modify the /etc/hosts file:

```
#host node1_boot
node1_boot1 is 192.168.10.101
```

4. Check the serial network connection:
  - a. Check that both machines are connected by RS232 cable.
  - b. Configure tty device on machine #1 and machine #2 using **smitty**:

**# smitty tty**

Select **Add a tty** → **tty rs232 Asynchronous Terminal** → **sa0 Available 01-S1 Standard I/O Serial Port**. In **Add a TTY** screen (Figure 10-7), press F4 to show the list of available port numbers. Select the port number displayed in the list.

Add a TTY		
[TOP]	[Entry Fields]	
TTY type	tty	
TTY interface	rs232	
Description	Asynchronous Terminal	
Parent adapter	sa0	
* PORT number	[0]	+
Enable LOGIN	disable	+
BAUD rate	[]	+
PARITY	[none]	+
BITS per character	[8]	+
Number of STOP BITS	[1]	+
TIME before advancing to next port setting	[0]	+#
TERMINAL type	[dumb]	
FLOW CONTROL to be used	[xon]	+
OPEN DISCIPLINE to be used	[dtropen]	+
[MORE...30]		

Figure 10-7 Add a TTY

- c. Check that the tty device is available as shown here:

**# lsdev -Cc tty**

tty0 Available 01-S1-00-00 Asynchronous Terminal

## Configuring HACMP

AIX **smitty** provides HACMP configuration interfaces. The main steps for this are as follows:

1. Add nodes to an HACMP cluster.
2. Add a service IP label/address.
3. Configure application servers.
4. Add a resource group.
5. Add resources to the resource group.
6. Configure persistent IP alias for HADR communication.
7. Define the serial network and the serial network device.
8. Verify and synchronize HACMP configurations.
9. Conduct a basic verification of HACMP configuration.

Next we provide the details of the steps to be followed:

- 1. Add nodes to an HACMP cluster:

**#smitty hacmp**

From the HACMP for AIX menu, select **Initialization and Standard Configuration → Add Nodes to an HACMP Cluster**

In Configure Nodes to an HACMP Cluster (standard) menu (Figure 10-8), enter the cluster name and new nodes.

Configure Nodes to an HACMP Cluster (standard)

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

\* Cluster Name

[hadr\_cluster]

New Nodes (via selected communication paths)

[node1\_boot1 node2\_boot1]

Currently Configured Node(s)

Figure 10-8 Add node to HACMP cluster

- 2. Add a service address to the resource group:

**# smitty hacmp**

From HACMP for AIX menu, select **Initialization and Standard Configuration → Configure Resources to Make Highly Available → Configure Service IP Labels/Addresses → Add a Service IP Label/Address.**

In the Add a Service IP Label/Address (standard) menu, enter the IP label and the network name as shown in Figure 10-9.

Add a Service IP Label/Address (standard)

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

\* IP Label/Address

[service ]

\* Network Name

[net\_ether\_01]

Figure 10-9 Add a service IP label



### 3. Prepare the application server start/stop scripts.

In this example, we use dummy start and stop scripts just for testing the HACMP setup. We provide the full functional start and stop scripts to control HADR database in an HACMP environment in the next section.

Example 10-3 is a dummy start/stop script which just records the time when the scripts are executed. We place the scripts in /home/hadrinst/scripts directory. Copy this dummy script and change its name as follows:

- Start script of HACMP application server:  
/home/hadrinst/scripts/hadr\_primary\_takeover.ksh
- Stop script of HACMP application  
server:/home/hadrinst/scripts/hadr\_primary\_stop.ksh

*Example 10-3 Dummy application server start/stop script*

---

```
#!/usr/bin/ksh -x

exec >> /tmp/~basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

exit 0
```

---

Make sure that these scripts have execution permissions:

```
#ls -l /home/hadrinst/scripts
```

### 4. Configure the application server.

To access this menu, complete the following steps:

```
# smitty hacmp
```

From HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure Resources to Make Highly Available** → **Configure Application Servers** → **Add an Application Server**.

In the Add Application Server menu, enter the server name, and complete the paths of the start and stop scripts, as shown in Figure 10-10.

Add Application Server

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

\* Server Name

[hadr\_server ]

\* Start Script

[/home/hadrinst/scripts/hadr\_primary\_takeover.ksh ]

\* Stop Script

[/home/hadrinst/scripts/hadr\_primary\_stop.ksh ]

Figure 10-10 Adding application server

5. Add a resource group.

To add a resource group, go to **smitty**:

**# smitty hacmp**

From HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Add a Resource Group**.

In the Add a Resource Group menu, enter the Resource group name and the participating node names, as shown in Figure 10-11. This corresponds to *Rotating resource group* in the earlier HACMP versions, which means there is no priority for resource group between nodes. For further details, see *HACMP for AIX 5L V5.3 Planning and Installation Guide*, SC23-4861-08, Chapter 6.

Add a Resource Group

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

\* Resource Group Name

[hadr\_rg ]

\* Participating Node Names (Default Node Priority)

[node1 node2 ]

+

Startup Policy

Online Using Distribution Policy

+

Fallover Policy

Fallover To Next Priority Node in The List

+

Fallback Policy

Never Fallback

+

Figure 10-11 Adding a resource group

6. Add resources to the resource group:

# smitty hacmp

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Configure HACMP Resource Groups** → **Change/Show Resources for a Resource Group (standard)**.

In the Change/Show Resources for a Cascading Resource Group menu, enter service IP label/address, service, as shown in Figure 10-12.

Change/Show Resources for a Cascading Resource Group

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

[Entry Fields]

Resource Group Name

hadr\_rg

Participating Node Names (Default Node Priority)

node1 node2

Startup Policy

Online Using Distribution Policy

Fallover Policy

Fallover To Next Priority Node In The List

Fallback Policy

Never Fallback

Service IP Labels/Addresses

[service ]

+

Application Servers

[hadr\_server]

+

Volume Groups

[ ]

+

Use forced varyon of volume groups, if necessary

false

+

Filesystems (empty is ALL for VGs specified)

[ ]

+

Figure 10-12 Adding resource to resource group

## 7. Define serial N/W and serial network device.

The subsidiary network for HACMP keep-alive is recommended to make HACMP failure detection more secure. For more details, see *HACMP 5.4 Administration Guide*, SC23-4862-09. Enter:

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Extended Configuration** → **Extended Topology Configuration** → **Configure HACMP Communication Interfaces/Devices** → **Add Communication Interfaces/Devices** → **Add Discovered Communication Interface and Devices** → **Communication Devices**.

In the Select Point-to-Point Pair of Discovered Communication Devices to Add menu, choose the tty device on each node you connected with RS-232C cable. In our example, we select node1 tty2 and node2 tty2.

## 8. Verify and synchronize HACMP configurations.

After defining the HACMP configuration from one node, you must verify and synchronize the cluster topology to the other node.

```
# smitty hacmp
```

From the HACMP for AIX menu, select **Initialization and Standard Configuration** → **Verify and Synchronize HACMP Configuration**.  
The HACMP verification utility checks that the cluster definitions are the same on all the nodes and provides diagnostic messages if errors are found.

**Starting HACMP**

HACMP configuration is now complete. The next step is to start up the HACMP cluster as follows:

- 1. Start HACMP on service node Node1.  
First you start up HACMP on service node Node1. To start up HACMP from the **smit** menu, complete these steps:  
**# smitty clstart**  
In Start Cluster Services menu, select **now** on Start now, on system restart or both; select **true** on Startup Cluster Information Daemon?; leave the default for all the other fields, as shown in Figure 10-13.

Start Cluster Services

Type or select values in entry fields.  
Press Enter AFTER making all desired changes.

[Entry Fields]

* Start now, on system restart or both	now	+
Start Cluster Services on these nodes	[node1 ]	+
BROADCAST message at startup?	false	+
Startup Cluster Information Daemon?	true	+
Reacquire resources after forced down ?	false	+

Figure 10-13 Starting cluster services

Check cluster.log to see if HACMP is started successfully by using the following command:  
**# tail -f /usr/es/adm/cluster.log**  
The message **EVENT COMPLETED: node\_up\_complete xxx** indicates that HACMP is started successfully, where xxx is the node number. Figure 10-14 shows the output of cluster.log.

```
Oct 17 20:04:00 node1 HACMP for AIX: EVENT START: node_up node1
Oct 17 20:04:02 node1 HACMP for AIX: EVENT START: acquire_service_addr
Oct 17 20:04:05 node1 HACMP for AIX: EVENT START: acquire_aconn_service en0 net_ether_01
Oct 17 20:04:05 node1 HACMP for AIX: EVENT COMPLETED: acquire_aconn_service en0 net_ether_01 0
Oct 17 20:04:05 node1 HACMP for AIX: EVENT COMPLETED: acquire_service_addr 0
Oct 17 20:04:06 node1 HACMP for AIX: EVENT COMPLETED: node_up node1 0
Oct 17 20:04:08 node1 HACMP for AIX: EVENT START: node_up_complete node1
Oct 17 20:04:09 node1 HACMP for AIX: EVENT START: start_server hadr_server
Oct 17 20:04:10 node1 HACMP for AIX: EVENT COMPLETED: start_server hadr_server 0
Oct 17 20:04:10 node1 HACMP for AIX: EVENT COMPLETED: node_up_complete node1 0
```

Figure 10-14 Cluster.log output

2. Start HACMP on standby node Node2.

After the HACMP service node is started, you can use the same procedure to start the standby node.

After HACMP is started on both the nodes, check the following items:

- ▶ Check the status of the resource group using the HACMP command: **clRGinfo**. You can issue this command on either node. Example 10-4 shows the out put of this command. You can see that resource group hadr\_rg is OFFLINE on node1 and ONLI NE on node2, which means that the resource group hadr\_rg is now acquired by node2.

**# /usr/es/sbin/cluster/utilities/clRGinfo**

Example 10-4 Check resource group status

```
# /usr/es/sbin/cluster/utilities/clRGinfo
```

Group Name	State	Node
hadr_rg	OFFLINE	node1
	ONLINE	node2

- ▶ Check if the service address is added on service node Node1.

**#netstat -i | grep -v link**

Example 10-5 shows the output of the **netstat** command. The service IP should be in the output list.

#### Example 10-5 Check service address

---

```
root@node1:/home/hadrinst/scripts# netstat -i | grep -v link
Name Mtu Network Address IpKts Ierrs OpKts Oerrs Coll
en0 1500 192.168.10 node1_boot1 3268881 0 2956143 0 0
en0 1500 9.43.86 service 3268881 0 2956143 0 0
en1 1500 192.168.20 node1_boot2 793555 0 1671197 0 0
```

---

- On the service node, make sure that the application server is started.

When the HACMP is started, the application server Start script will be executed. Check the log written by the Start script. In our example, it is `hadr_primary_takeover.ksh.log`.

```
# cat /tmp/hadr_primary_takeover.ksh.log
```

Example 10-6 shows the sample output. Since now we have just set a dummy script, nothing happens on HADR databases. The script only records the time when the script is issued to the log file.

#### Example 10-6 Check application server

---

```
hadr_rg:[6] echo
#####
#####
hadr_rg:[7] date
Tue Oct 17 20:04:10 CDT 2006
```

---

### HACMP takeover test

You can test the HACMP takeover function by stopping HACMP on the service node in takeover mode. The standby node should take over the resource group after the takeover operation. To stop HACMP in the takeover mode, use **smitty** in service node, Node1 in our example:

```
#smitty clstop
```

In Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node where HACMP will be stopped; select **takeover** on Shutdown mode. See Figure 10-16.

Stop Cluster Services

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

[Entry Fields]

\* Stop now, on system restart or both

now

+

Stop Cluster Services on these nodes

[node1 ]

+

BROADCAST cluster shutdown?

false

+

\* Shutdown mode

takeover

+

Figure 10-15 Stop HACMP with takeover

- ▶ Check whether the HACMP takeover process has stopped the service node.  
The HACMP takeover process writes a message to cluster.log. The node\_down\_complete xxx event message in the log indicates that the takeover event is now complete in the service node, Node1as shown in Example 10-7.  
**# tail -f /usr/es/adm/cluster.log**

Example 10-7 Check takeover result

```
# tail -f /usr/es/adm/cluster.log
.....
Oct 20 20:59:55 node1 HACMP for AIX: EVENT COMPLETED: node_down_complete node1
.....
```

- ▶ Check that the application server is stopped on service node Node1.  
View the application server stop script log to see whether the application server has been stopped successfully:  
**# cat /tmp/hadr\_primary\_stop.ksh.log**
- ▶ On the service node Node1, check if the service address is released:  
**# netstat -i**
- ▶ On standby node Node2, check that node\_down\_complete Node1 event is completed:  
**# tail -f /usr/es/adm/cluster.log**



Example 10-8 shows a snippet of the cluster.log.

*Example 10-8 Check node\_down\_complete event from Node 2*

```
# tail -f /usr/es/adm/cluster.log
.....
Oct 20 20:59:55 node2 HACMP for AIX: EVENT COMPLETED: node_down_complete node1
.....
```

- Check that Node2 owns the resource group using the `clRGinfo` command as shown in Example 10-9.

*Example 10-9 Checking if Node2 has taken over the resource*

```
# /usr/es/sbin/cluster/utilities/clRGinfo
-----
Group Name      State              Node
-----
hadr_rg         OFFLINE            node1
                 ONLINE             node2
```

- Check that Node2 has taken over the service IP address. Issue the `netstat` command in Node2 as shown in Example 10-10.

*Example 10-10 Check service IP address*

```
root@node2:/home/hadrinst/scripts# netstat -i | grep -v link
Name Mtu Network Address IpKts Ierrs OpKts Oerrs Coll
en0 1500 192.168.10 node2_boot1 3268881 0 2956143 0 0
en0 1500 9.43.86 service 3268881 0 2956143 0 0
en1 1500 192.168.20 node2_boot2 793555 0 1671197 0 0
```

- Check that the application server is started on Node2.  
Since our start/stop scripts are still dummy, HADR takeover is not issued yet. Here we just checked if HACMP functions work correctly:

```
# cat /tmp/hadr_primary_takeover.ksh.log
```

## Stopping HACMP

To stop HACMP, use `smitty` on both Node1 and Node 2.

```
#smitty clstop
```

In the Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node to be stopped; and select **graceful** for Shutdown mode, as shown in Figure 10-16.

Stop Cluster Services

Type or select values in entry fields.

Press Enter AFTER making all desired changes.

[Entry Fields]

\* Stop now, on system restart or both

now

+

Stop Cluster Services on these nodes

[node1 ]

+

BROADCAST cluster shutdown?

false

+

\* Shutdown mode

graceful

+

Figure 10-16 Stop HACMP

### 10.2.5 Prepare application server scripts

To automate HADR takeover with HACMP, we utilize the HACMP application server start/stop scripts. By including the HADR commands in the start/stop scripts, HACMP can handle HADR operations in connection with the resource group. The application scripts for HADR can be placed in the directory of your choice. We place our scripts in the /home/hadrinst/scripts directory. The sample scripts can be found in Appendix A, “HACMP application server scripts” on page 403. Here we explain some of these HACMP controlled scripts and how they work.

#### ***hadr\_primary\_takeover.ksh (start script)***

We define a shell script *hadr\_primary\_takeover.ksh* as the start script for HACMP application server. This script is executed in the following situations when the node acquires the resource group:

- ▶ Normal start up of HACMP on the first (service) node in the cluster.
- ▶ Unplanned takeover that is triggered by failure detection of HACMP.
- ▶ Planned takeover by stopping HACMP in takeover mode, or moving the resource group by an administrative command.

This script is executed not only when the resource group has fallen over to the standby node, but also when the first node in the cluster is started and acquires the resource group. This script checks which is the starting trigger by issuing the HACMP command: **c1RGinfo**.

Figure 10-17 shows the flow chart of the script, *hadr\_primary\_takeover.ksh*. If the trigger is normal HACMP start up, then this script simply exits. If the starting trigger is takeover, which means that this script is issued on the standby node in connection with the resource group fall over, then this script checks the HADR role. If the HADR role is standby, it issues the **takeover hadr** command with *by force* option.

In circumstances where the standby database cannot communicate with the primary database, the **takeover hadr** command needs the *by force* option to change its role to primary. But for planned takeover, *by force* option is not preferable, so in our example, the normal **takeover hadr** command (without the *by force* option) is issued from the stop script by a remote command before this script is issued on the standby node.

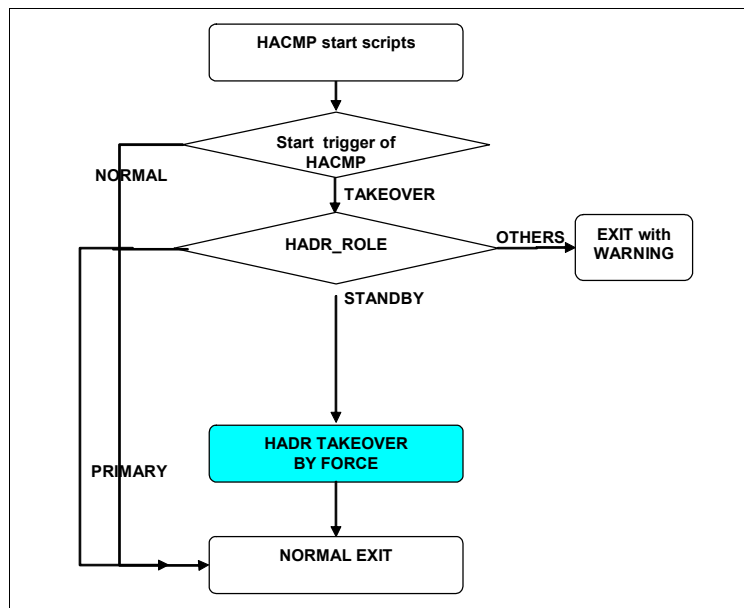


Figure 10-17 *hadr\_primary\_takeover.ksh* script flow

### ***hadr\_primary\_stop.ksh* (stop script)**

We define *hadr\_primary\_stop.ksh* as the stop script for HACMP application server. Figure 10-18 shows the script logic flow. This script is executed on the service node and when the node releases the resource group. That is, when you stop HACMP in takeover mode or move the resource group from one server to the other intentionally for planned takeover. For planned takeover, it is preferable to issue the **takeover hadr** command as soon as possible. In our example, this script issues a remote command to the standby node to execute the **takeover hadr** command.

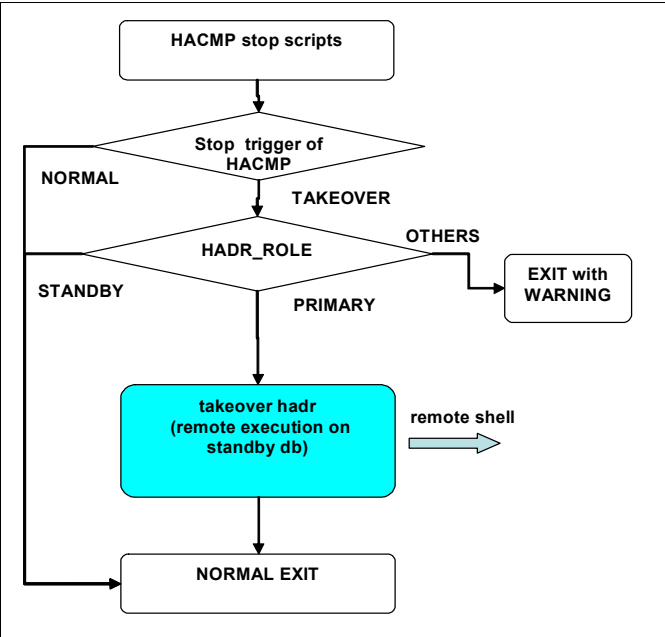


Figure 10-18 hadr\_primary\_stop.ksh script flow

**Additional information for HACMP commands**

You can use the output of the HACMP command **c1RGinfo** to define the start trigger of the application server:

```
/usr/es/sbin/cluster/utilities/c1RGinfo -a
```

In the Resource Group Movement column, you can see just one node when the start trigger is normal startup (Example 10-11); two nodes when the start trigger is takeover (Example 10-12).

*Example 10-11 Normal start up on node1*

```
# /usr/es/sbin/cluster/utilities/c1RGinfo -a
```

Group Name	Type	Resource Group Movement
hadr_rg	non-concurrent	PRIMARY=" :node1"

*Example 10-12 Takeover from node1 to node2*

```
# /usr/es/sbin/cluster/utilities/c1RGinfo -a
```

Group Name	Type	Resource Group Movement
------------	------	-------------------------

---

```
hadr_rg          non-concurrent PRIMARY="node1:node2"
```

---

## 10.2.6 Joint test for HADR and HACMP

After the functional scripts have replaced the dummy scripts, we can test the combination of HADR automatic takeover in an HACMP environment. We implement the joint test for the following scenarios:

► **Planned failover:**

In the first scenario, we use the facility provided by the HACMP application to test the failover between the two nodes.

► **Unplanned failover:**

For the unplanned failover scenario, we halt the primary system to simulate a real system outage and verify that the failover occurs properly.

### Start up HADR and HACMP in the cluster

To carry out the test scenarios, complete the following steps to start up HADR and HACMP:

1. Start standby database on node2.

Start up the HADR databases you have already set up in 10.2.3, “HADR setup” on page 263. Start the standby database on Node2 first:

- Check that HADR role is standby on node2:

```
$db2 get db cfg for sample
```

- Start up the standby database:

```
$db2 activate db sample
```

**Tip:** When you start up the HADR process, you do not have to issue **start hadr** or **stop hadr** command every time. Once you start an HADR database in a primary or standby role, just activate and deactivate database can be used to start and stop the HADR database.

2. Start the primary database on Node1:

- Check that HADR role is primary on node1:

```
$db2 get db cfg for sample
```

- Start up the standby database:

```
$db2 activate db sample
```

- Check if HADR status is in the Peer state:  
`$db2pd -hadr -db sample`
- 3. Start HACMP on Node1.  
 For details, See “Starting HACMP” on page 271.
- 4. Start HACMP on Node2.  
 For details, See “Starting HACMP” on page 271.
- 5. Check HADR and HACMP status and they are ready for the test:
  - Check if HADR is in the Peer state:  
`$db2pd -hadr -db sample`
  - Check if resource group is ONLINE on node1:  
`# /usr/es/sbin/cluster/utilities/c1RGinfo`

## Configure client node

To check the client access after takeover, configure the client machine by completing the steps listed below. We configured the client machine on Node3.

1. Create instance on client node:  
`$db2icrt -u clntinst clntinst`
2. Catalog node directory specifying the service address: service  
`$db2 catalog tcpip node SERVICE remote service server 50030`
3. Catalog database:  
`$ db2 catalog database sample at node SERVICE`
4. Connect to database server:  
`$ db2 connect to sample user hadrinst using hadrinst`
5. Type the query shown in Example 10-13 to check which server you are connecting to. You can see that we are now connected to Node1.

*Example 10-13 Check the server connected*

---

```
$ db2 "select substr(host_name,1,8) as hostname from table
(db_partitions()) as t"
```

```
HOSTNAME
```

```
-----
```

```
node1
```

```
1 record(s) selected.
```

---

## Planned failover and fallback

In the first scenario, we use the facility provided by the HACMP application to test the failover between the two nodes:

1. Stop HACMP in takeover mode on service node, Node1:

```
#smitty clstop
```

In Stop Cluster Services menu, select **now** on Stop now, on system restart or both; select the node where HACMP will be stopped; select **takeover** on Shutdown mode.

2. Check if HACMP takeover process has stopped the service node.

The HACMP takeover process writes a message to cluster.log. The node\_down\_complete xxx event message in the log indicates that the takeover event is completed in the service node, Node1.

```
# tail -f /usr/es/adm/cluster.log
```

3. Check that the **stop** script is issued on Node1 and the TAKEOVER HADR command was issued remotely on Node2. You will see the output as in Example 10-14 in the log file of this script.

```
#cat hadr_primary_stop.ksh.log
```

*Example 10-14 Output of stop script*

---

```
....
HADR takeover is issued on remote node
.....
node2: DB20000I The TAKEOVER HADR ON DATABASE command completed
successfully.
```

---

4. Check the current status of HACMP on Node2. The resource group should be taken over to Node2. See Example 10-15.

*Example 10-15 Check who owns resource group*

---

```
# /usr/es/sbin/cluster/utilities/clRGinfo
```

Group Name	State	Node
hadr_rg	OFFLINE	node1
	ONLINE	node2

---

5. Check the status of the HADR database on both the nodes. Now the primary database is running on Node2 and the standby database is running on Node1:

```
$db2pd -hadr -db sample
```

6. Reconnect from the client node, Node3, and issue the query as shown in Example 10-16. You can see that you have connected to node2. In this case, you must reconnect to the server after you receive a communication error. If the alternate server is set, reconnection is automatically done by Automatic Client Reroute.

*Example 10-16 Check reconnected node*

---

```
$ db2 "select substr(host_name,1,8) as hostname from table
(db_partitions()) as t"
```

```
HOSTNAME
```

```
-----
```

```
node2
```

```
1 record(s) selected.
```

---

7. Start HACMP on Node1 again. Check that there is no change in the HADR status and the resource group is still ONLINE on Node2.
8. Fallback from Node2 to Node1:  
To fallback the resource group from Node2 to Node1, you can stop HACMP in takeover mode on Node2.
9. Start HACMP on Node2: The primary database is running on Node1 and the standby database is running on Node2. Now the cluster has returned to the state we started with.

## Unplanned failover for system crash

In this scenario, we test the failover by halting the system intentionally. We check that HACMP detects the failure and that the standby database is switched to the primary database in connection with the HACMP resource group takeover.

1. Check the HACMP and HADR status of the two nodes:
  - HACMP service is started on both nodes.
  - HADR is in the Peer state.
2. Connect from the client and issue some queries following the steps shown in “Configure client node” on page 280.
3. On Node1, issue the following command as root:

```
#sync;sync;sync;halt -q
```

Node1 will terminate processing immediately. Soon HACMP detects the outage and starts failover of the resource group to Node2.

4. Check that the resource group is taken over by Node2:

```
# /usr/es/sbin/cluster/utilities/c1RGinfo
```



5. Check that the start script is issued on Node2.

```
# view /tmp/hadr_primary_takeover.ksh.log
```

6. Check that the primary HADR database is now on Node2

```
$ db2pd -hadr -db sample
```

If the HADR primary database is on Node2, the failover is successful.

Next, we want to failback the resource group to the original node, as the failed system situation has been repaired. Power on Node1 and start DB2 instance.

```
$db2start
```

You need to reintegrate the old primary database to the current state using the HADR function. In the old primary system (Node1) do the following actions:

1. Check the HADR role on Node1: The HADR role on Node1 should still show as primary. See Example 10-17.

*Example 10-17 Check HADR role*

---

```
$ db2 get db cfg for sample | grep "HADR database role"
```

HADR database role	= PRIMARY
--------------------	-----------

---

2. Check the db2diag.log file using the following command to see the HADR catching up process. After you start the HADR database as standby on Node1, the catching up process will start.

```
$ db2diag -f
```

or

```
$ tail -f /home/hadrinst/sql1lib/db2dump/db2diag.log
```

3. Restart the HADR database as standby on Node1.

To reintegrate the database on Node1 into HADR pair as a standby database, complete the following steps:

- a. To switch the role of the database from primary to standby, you have to issue the **start hadr** command instead of **activate database**. See Example 10-18. Check the messages in the db2diag.log until they return to the peer status.

*Example 10-18 Start database as HADR standby*

---

```
$db2 "start hadr on db sample as standby"
```

```
DB20000I The START HADR ON DATABASE command completed successfully.
```

---

- b. Check whether the HADR status has come back to the Peer state after catching up the log from the primary on Node1 as follows:
  - **db2pd** command shows the current HADR status.
  - In the db2diag.log file you will see messages as shown in Example 10-19.

*Example 10-19 db2diag.log message*

---

```

2006-10-30-23.49.47.617235+540 E353590C344          LEVEL: Event
PID       : 43462                TID  : 1          PROC : db2hadrs (SAMPLE) 0
INSTANCE: hadrinst              NODE  : 000         DB   : SAMPLE
FUNCTION: DB2 UDB, High Availability Disaster Recovery, hdrSetHdrState,
probe:10000
CHANGE   : HADR state set to S-Peer (was S-NearlyPeer)

```

---

4. Start HACMP on Node1 again. Check that there is no change in the HADR status and the resource group is still ONLINE on Node2.
5. Fallback from Node2 to Node1: To fallback the resource group from Node2 to Node1, you can stop the HACMP in takeover mode on Node2.
6. Start HACMP on Node2.

After the primary database is running on Node1 and the standby database is running on Node2, you can start HACMP on Node2. The cluster now returns to the state we started from.

## Stop HACMP and HADR

Complete the following steps to stop the HACMP system for a maintenance operation:

1. Stop HACMP on the Service node and the standby node.  
See “Stopping HACMP” on page 275.
2. Stop HADR:
  - Stop HADR on the primary database on the service node:  
\$db2 deactivate db sample
  - Stop instance on the service node:  
\$db2stop
  - Stop HADR on the standby database on the standby node:  
\$db2 deactivate db sample
  - Stop instance on the standby node:  
\$db2stop

**Note:** The **deactivate database** command can be used to stop the HADR process instead of the **stop hadr** command. When you **deactivate database**, the HADR database role stays in database configuration. If you issue the **stop hadr** command, the HADR database role changes to STANDARD, which makes it hard for you to know which node was primary and which node was standby.

## 10.3 Automating HADR takeover with TSA

IBM Tivoli System Automation for Multiplatforms (TSA) provides a framework to automatically manage the availability of what are known as resources. Examples of resources includes:

- ▶ Any piece of software for which start, monitor, and stop scripts can be written to control.
- ▶ Any Network Interface Card (NIC) for which TSA has been granted access to. That is, TSA will manage the availability of any IP address that a user wishes by floating that IP address amongst NICs that it has been granted access to.

For example, both a DB2 instance and HADR itself, have start, stop, and monitor commands. Therefore, TSA scripts can be written to automatically manage these resources. Scripts, as well as other attributes of a resource, are needed by TSA to manage that resource. TSA stores a resource's attributes in an object container, much like the attributes of a Java class. In fact, TSA manages a resource by instantiating a class for that resource.

TSA also allows related resources to be managed in what are known as resource groups. TSA guarantees that all resources within a given resource group will be online at one and only one physical node at any point in time. Also, all of those resources will reside on the same physical node. Examples of resource groups (such as related resources) are a DB2 instance, its IP address, and all of the databases that it manages.

Finally, TSA provides high availability (HA), for any resource group that it manages, by restarting all of its resources if it fails. The resource group will be restarted on an appropriate node in the currently online cluster domain. An appropriate node must contain a copy of all of the resources, that are defined in the failing resource group, to be selected as a node to restart on.

## TSA components

Since V2.1 (V2.2 at the time of writing), TSA consists of these components:

► *End-to-end automation component:*

This is the second-tier-enabling technology, which gives TSA a rare ability to provide control over heterogeneous environments of multiple base-level clusters. The low level clustering software, currently installed, is not replaced. TSA end-to-end automation can control an existing HACMP on AIX, Heartbeat on Linux, or MSCS just as easily as if it were controlling a TSA base component cluster on these same platforms. TSA software interfaces called *End-to-End Automation Adapters* operate between TSA and the base-level clustering software.

For more details, refer to:

*End-to-end Automation with IBM Tivoli System Automation for Multiplatforms*, SG24-7117.

► *Base component:*

This TSA component provides base-level cluster functionality. It acts as an equivalent to HACMP or other clustering software, on AIX and Linux. The TSA base component consists of the following parts:

a. *Automation adapter:*

At the base component level, a *first level automation adapter* is used to directly control resources on nodes inside the base-level cluster.

At the end-to-end automation component level, an *end-to-end automation adapter* provides an interface between the automation layer already existing within a single base-level cluster, and the end-to-end automation layer which spans multiple clusters.

b. *Operations console of the base component:*

The TSA Operations Console is a Web browser accessed feature (no client software install required), based on the WebSphere Portal Server Integrated Solutions Console. Because it spans all TSA domains, and provides the same look and feel, there is no interface discrepancy between platforms.

While the Operations Console is there to help monitor the automation and current states of various resources, it also allows manual resource state changes (for example if a server is required to be taken offline for maintenance), and performs takeovers accordingly. The Operations Console can operate in three different modes:

- i. *End-to-end automation mode:* Where end-to-end automation is essentially active.

- i. *First-level automation mode*: Where end-to-end automation has been installed, but is not active.
- ii. *Direct access mode*: Where you are essentially running the console directly on a system which is part of a base-level cluster. TSA end-to-end automation itself is not generally installed here. There is just an automation adapter plug-in on base-level cluster nodes for interfacing to non-TSA clustering software.

For more details, refer to:

*Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide, SC33-8272.*

## Understanding automation on TSA

The following are the common terms between TSA and other clustering solutions:

- ▶ *Resource*: This can be a physical device such as: a disk, a network adapter; or software, from the Operating System itself, through middle ware and DBMS, to WebSphere Administration Server (WAS) and Web Server daemons. With respect to up/down status monitoring and automation (and now end-to-end automation), resources can also exist at multiple tiers, for example, grouping granular resources and base-level cluster environments to make the concept of an application environment.
- ▶ *Critical resource*: This is a resource that must not be active on more than one node at the same time. A primary example of this is a static IP address that is shared between the nodes which clients use to connect to server resources. By this definition, the HADR primary role database would also be a critical resource.
- ▶ *Cluster/peer domain*: This is a group of host systems or nodes. These are what resources are part of, so if groups of resources become unavailable, it essentially indicates that a given node, or subcluster is unavailable. This leads to automated actions designed to maintain availability by establishing *quorum*, and to attempt restoration of the failed nodes without conflicting with resources on available nodes.
- ▶ *Quorum*: The majority level of operational nodes, required before certain operations can be performed within a cluster. TSA splits this definition into:
  - *Configuration quorum*: Determines whether cluster configuration changes will be accepted.
  - *Operational quorum*: Determines whether state changes of resources will be accepted (so as not to cause conflicts).

- ▶ *Tie-breaker*: Where equal number of nodes exist in a subcluster, this determines which of those subclusters can have the quorum. The mechanism of the tie-breaker can be of six different types, depending on the platform and devices available:
  - *Operator*: Requires human intervention for a decision in order to reach a quorum.
  - *Fail*: A default/pseudo tie-breaker in which neither subcluster attains quorum.
  - *SCSI*: Shared SCSI disk implementation on certain Linux platforms. SCSI reserve or persistent reserve command is used to achieve tie-breaker exclusive status.
  - *ECKD™*: Shared ECKD disk on system z Linux. ECKD reserve command is used to achieve tie-breaker exclusive status
  - *Disk*: Shared SCSI-like disk on AIX. An equivalent or pseudo-SCSI reserve or persistent reserve command is used to achieve tie-breaker exclusive status on SCSI-like or emulated disk.
  - *EXEC*: Network implementation which calls a custom executable to achieve a RSCT (Reliable Scalable Cluster Technology) exec tie-breaker. ICMP (Internet Control Message Protocol) ping echo requests are sent to an external IP instance to determine tie-breaker exclusive status in the case of subcluster nodes failing. Note that this method relies on failure of ICMP response to indicate subcluster/node failure and is unable to determine if communication is still working between the subcluster peer nodes.

In the terminology specific to TSA, the hierarchy by which Tivoli System Automation coordinates resources in a clustered environment comprises an *Automation Manager*, which consists of a *Binder* and a *Logic Deck*.

- ▶ *Binder*: Has resources assigned (bound) to a particular node. If the resource cannot be accessed on any node, then it is placed in a *Sacrificed* state. If it can be accessed, then it is placed in a *bound* state to that node. Resources in a bound state are only actually started if resource dependencies have been met.

If System Automation has not yet tried to access a given resource, it is by default in an *unbound* state. Conflicts between multiple available resources are managed by a priority value. The resource with lower priority is placed in *Sacrificed* state.

- ▶ *Logic deck*: Sends orders to start and stop resources according to the rules set out by the *Automation Policy*.
- ▶ *Automation Policies*: An abstract business rule-based set of dependencies and relationships between resources, rather than hard and fast scripts. The

Automation Policies determine what should occur when values change in the monitored states of resources, specifically when differences occur between a *Desired State*, (Online/Offline), and an *Observed State*

- ▶ *Resource relationships*: A critical part of the automation policy
  - *Stop-start relationships* determine which resources should be stopped or started in a given scenario.
  - *Location relationships* determine whether related resources should be started on the same or on separate nodes.
- ▶ *Equivalency*: A group of resources which can be said to provide the same function: TSA can choose from any one of these in order to provide a given function in the required definition of availability. For example, given a number of redundant network adapters, if one fails, that IP address can be mapped by TSA to another adapter in the equivalency.

Technically, TSA Automation Policies themselves do not entail scripting. These policies are in the form of business rules or heuristics which dictate at a high level how resources on various nodes should behave and interact. This enables attribution of priorities such that for any given partial or total resource failure scenario, the appropriate action will be taken.

At a lower level, resources are started, stopped, and monitored using scripts or command line interfaces to the resource in question. Some scripts are provided with the TSA product for certain common resource types, and some are available as downloadable samples. Scripts would have to be written in-house for any uncommon resources or software with which TSA is not familiar.

Sample scripts for many TSA environments can be found at the Web site:

<http://www.ibm.com/software/tivoli/products/sys-auto-linux/downloads.html>

An example of a lower level script for resource interaction is the status script required for HADR on DB2 in a TSA-managed cluster environment. This could take the form of a shell script executing db2pd or get snapshot with some text/string manipulation logic for example. The scripts for some UNIX platforms are supported, but may have to be adapted for other platforms.

HADR status monitoring in DB2 9 is made extremely easy by simple table functions to retrieve the HADR monitor elements. Previously, a script had to interpret the output of db2pd, db2 get snapshot for database, or continuously monitor and interpret the output of db2diag.log Details can be found in 4.5, “Monitoring HADR: db2diag” on page 107. For efficiency and speed required in critical banking applications, it may be best to continue using a script tailored to continually tail the last several lines of db2diag.log to check on the

peer/connected state of HADR, and forward that to the standby in case of primary failure.

High Availability failover behavior for TSA is also set up using configuration scripts. For those who want to avoid as much script writing as possible, a reference article for simplifying the configuration of failover behavior can be found at the Web site:

<http://www.ibm.com/developerworks/ibm/library/i-odoebp16/>

That article can be particularly valuable to anyone using a similar environment to the article's test platform of an application running on a remote Network File System (NFS) mount, IBM HTTP Web Server, J2EE™, WebSphere Administration Server (WAS), and DB2.

Understanding the interaction between the components within this hierarchy is beyond the scope of this book. The reference material for TSA is as follows:

- ▶ *Tivoli System Automation for Multiplatforms Version 2.1, program number 5724-M00 - Base Component User's Guide, SC33-8210-05.*
- ▶ *Tivoli System Automation for Multiplatforms Version 2 Release 2 - Base Component Administrator's and User's Guide, SC33-8272.*
- ▶ *Tivoli System Automation for Multiplatforms Version 2.2 - Installation and Configuration Guide, SC33-8273-00.*
- ▶ *Tivoli System Automation for Multiplatforms Version 2.2 - End-to-End Automation Management Component Administrator's and User's Guide, SC33-8275-00.*
- ▶ *End-to-end Automation with IBM Tivoli System Automation for Multiplatforms, SG24-7117.*
- ▶ Tivoli System Automation V2.2 manuals can be found at the Web site:  
<http://publib.boulder.ibm.com/tividd/td/IBMTivoliSystemAutomationforMultiplatforms2.2.html>
- ▶ TSA introduction Web page:  
<http://www.ibm.com/software/tivoli/products/sys-auto-linux/index.html>

### 10.3.1 HADR with TSA<sup>1</sup>

To automate HADR takeover with Tivoli System Automation (TSA), in summary, DB2 HADR requires that TSA be configured with scripts to stop, start, takeover,

---

<sup>1</sup> Portions of this section are excerpted from the article, *Automating DB2 HADR Failover on Linux using Tivoli System Automation for Multiplatforms* by Steve Raspudic, Melody Ng, and Chris Felix, originally published as an IBM white paper, April 2006. [ftp://ftp.software.ibm.com/software/data/pubs/papers/hadr\\_tsa.pdf](ftp://ftp.software.ibm.com/software/data/pubs/papers/hadr_tsa.pdf)



and determine various status values for DB2 and HADR components, as well as automation policies which determine which actions to take in given scenarios or resource states.

Setting up HADR in conjunction with TSA is done by first setting up two servers (using Linux in the article as an example). These will become the primary and standby nodes within TSA, and will likewise contain the DB2 HADR primary and standby databases, respectively. A separate third server, (for example a client application server) can either then serve as a tiebreaker node for TSA, or a network tiebreaker can be set up which does not require a third server. Network tiebreaker support is described in further detail in *IBM Tivoli System Automation for Multiplatforms Base Component User's Guide*, SC33-8272-00.

In this section, we demonstrate the procedure to set up HADR with TSA using a typical topology as illustrated in Figure 10-19. In this figure, the primary node is hadr01, the standby node is hadr02, and the client application node is hadr03. TSA monitors the HADR pair for primary database failure, and will issue appropriate takeover commands on the standby database in the event of a primary database failure. Also, in addition to being an application server, hadr03 acts as a heartbeat node. That is, TSA on hadr03 will provide quorum between the primary and standby nodes, so as to prevent the scenario of having two primary databases (also known as the “split-brain” syndrome) in the event of simple network communications problems between hadr01 and hadr02.

For TSA cluster domains that are running TSA v2.1 or later, it is possible to use a network tiebreaker to provide quorum support. In this environment, only nodes the primary and the standby are required, with quorum support provided by a network tiebreaker resource, which can be created by following the chapter on “Protecting your resources – quorum support” in the document *IBM Tivoli System Automation for Multiplatforms Base Component User's Guide*, SC33-8272-00.

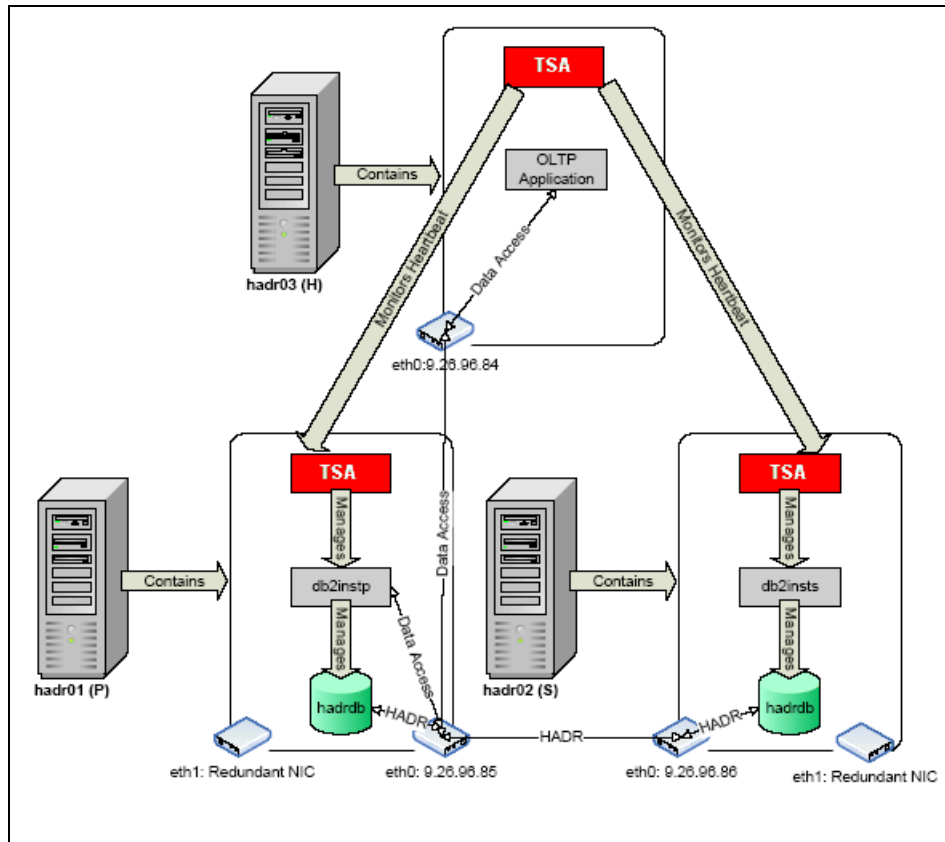


Figure 10-19 Typical HADR with TSA environment

## Software and hardware requirements

The minimum software requirement for running DB2 UDB on Linux can be found at the following Web site:

<http://www.ibm.com/db2/linux/validate>

For information about software requirements for running TSA, refer to:

<http://www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html>

Here is the actual software configuration used to set up the lab environment:

- ▶ Operating system: SuSE Linux Enterprise Server 9 Support Package 1 (kernel version 2.6.5-7.139-default)
- ▶ DB2 UDB product: DB2 UDB Enterprise Server Edition (ESE) Version 8.2.3
- ▶ Tivoli product: TSA V1.2 at FixPak 6 level
- ▶ glibc: glibc-2.3.3

The minimum hardware requirements for implementing the solution described in this section are the same as those documented here:

- ▶ For DB2 UDB, refer to:  
<http://www.ibm.com/db2/udb/sysreqs.html>
- ▶ For TSA, refer to:  
<http://www.ibm.com/software/tivoli/products/sys-auto-linux/platforms.html>

We used this actual hardware configuration to set up our lab environment:

- ▶ Two machines, each with the following configuration:
  - Processors used: Intel Xeon® 2 CPU 2.80 GHz
  - Memory: 512 MB
  - Network adapters: Two Intel PRO/100 Ethernet Adapters
- ▶ One machine with the following configuration:
  - Processors used: Intel Xeon 2 CPU 2.80 GHz
  - Memory: 512 MB
  - Network adapters: One Intel PRO/100 Ethernet Adapters

### 10.3.2 Setting up HADR with TSA

This section documents the procedures to set up the topology depicted in Figure 10-19 on page 292. It is a three node topology, in which one node (hadr01) hosts the primary database (hadrdb) and a second node (hadr02) hosts its standby. The third node will host the client application, as well as provide quorum between the primary and standby nodes.

Your topology does not have to include redundant NICs (eth1 in Figure 10-19 on page 292). Redundant NICs allow for recovery from simple outages caused by primary NIC failure (eth0). For example, if eth0 on hadr01 in Figure 10-19 on page 292 were to go down for some reason, then the IP address that it was hosting (that is, 9.26.96.85) could be taken over by eth1. In fact, there is an opportunity in “Set up highly available IP addresses” on page 299 to make each DB2 instance’s (db2instp and db2insts) IP address highly available with TSA.

The letters in front of a command in the following steps designate which node(s) a command is to be issued on to properly set up the topology as shown in Figure 10-19 on page 292. The order of the letters also designates the order in which you should issue a command on each node:

- ▶ (P): Primary Database Node (hadr01)
- ▶ (S): Standby database node (hadr02)
- ▶ (H): Heartbeat node (hadr03)

The parameters given for commands in the example are based on the topology in shown in Figure 10-19 on page 292. You can change the values accordingly to match your specific environment. Also, a “\” in a command designates that the command text continues on the next line. Do not include the “\” when you issue the command.

## Basic network setup

Make sure that all three nodes (hadr01, hadr02, and hadr03) are able to communicate with each other through TCPIP protocol.

To set up the network, perform the following steps:

1. Set up the network using either static IP or DHCP. In our example, we use static IP with a subnet mask of 255.255.255.0 for each node.
2. If you chose to use static IP addresses, then add the IP addresses to the hostname mapping information in /etc/hosts file on each node (P), (S) and (H). Following is sample content of the /etc/hosts file:

```
On (P): 9.26.96.85 hadr01
On (S): 9.26.96.86 hadr02
On (C): 9.26.96.84 hadr03
```

Adding static IP address to hostname mappings to the hosts file removes the systems’ DNS servers as a single point of failure. Should DNS fail, the cluster systems can still resolve addresses of the other machines via the hosts file.

Note that adding this information on the client is not recommended because of the maintenance overhead of updating all clients’ hosts files, should the mappings change.

3. Test that you can ping from each node to all other nodes successfully by using the following commands:

```
(P) (S) (H) # ping hadr01
(P) (S) (H) # ping hadr02
(P) (S) (H) # ping hadr03
```

## RSH setup

Many of the TSA commands that you will be issuing in the following steps require RSH to be set up on all three nodes. RSH allows a user from one node to run commands on another remote node.

Use the following steps to set up RSH:

1. Find out if rsh and rsh-server packages are installed in on each node by logging in as root and issuing the following commands:

```
(P) (S) (H) # rpm -qa | grep rsh
(P) (S) (H) # rpm -qa | grep rsh-server
```

If rsh and rsh-server are not installed, you can use YaST in SLES9 to select and install these packages, or you can install the packages directly from one of the Linux installation CDs by issuing the following commands as root:

```
(P)(S)(H) # rpm -ivh rsh-0.17-17.i386.rpm
(P)(S)(H) # rpm -ivh rsh-server-0.17-17.i386.rpm
```

For Red Hat and other Linux distributions, use the appropriate utilities to install the rsh and rsh-server package.

2. Enable RSH on each node by issuing the following command as root:

```
(P)(S)(H) # chkconfig rsh on
```

For Red Hat, also add a line “rsh” to the end of the /etc/securetty file on each of (P), (S) and (H).

Sample contents of /etc/securetty on (P), (S), and (H):

```
Console
Vc/1
...
tty11
rsh
```

3. Start RSH on each node by restarting the Internet daemon (inetd) or extended Internet daemon (xinetd), depending on your Linux distribution, as follows:

```
(P)(S)(H) # cd /etc/init.d
(P)(S)(H) # inetd restart
```

or

```
(P)(S)(H) # xinetd restart
```

4. Verify that rsh service is indeed started by issuing the following commands as root:

```
(P)(S)(H) # chkconfig --list | grep rsh
```

You should see output similar to the following lines:

```
(P)(S)(H) # chkconfig --list | grep rsh
rsh: on
```

5. Configure RSH to allow the root user to issue remote commands on each node by adding the following lines to the file /root/.rhosts:

On (P), (S), and (H):

```
hadr01 root
hadr02 root
hadr03 root
```

6. Verify that RSH is working by issuing the following commands as root. If you see the directory listing of /root on the node you are “rsh-ing” into, then rsh is working:

```
(P)(S)(H) # rsh hadr01 ls
(P)(S)(H) # rsh hadr02 ls
(P)(S)(H) # rsh hadr03 ls
```

## Install DB2 UDB

Install DB2 UDB V8.2.3 ESE (also known as V8.1 at Fixpak 10 level) on the primary and standby nodes (hadr01 and hadr02). *Do not* create instances at this step.

1. As root, go to the directory where the DB2 install code exists:

```
(P)(S) # cd /<directory_containing_DB2_install_code>
```

2. Install DB2 UDB ESE V8.2.3 by issuing the following command, choose to install DB2.ESE:

```
(P)(S) # db2_install
```

## Install TSA

Install TSA V1.2, and then upgrade to Fixpak 6 (TSA V1.2.6) on all three nodes.

1. As root, go to the directory where the TSA 1.2 install code exists and run the TSA 1.2 installer as follows:

```
(P)(S)(H) # cd /<directory_containing_TSA_install_code>/i386
(P)(S)(H) # installSAM
```

2. As root, go to the directory where the TSA 1.2 Fixpak 6 install code exists and run the TSA 1.2 installer as follows:

```
(P)(S)(H) # cd /<directory_containing_TSA_FP6>/SAM1204/i386
(P)(S)(H) # installSAM
```

3. Add the following lines to /root/.bashrc file:

On (P), (S) and (H):

```
export CT_MANAGEMENT_SCOPE=2
```

On (P) and (S):

```
PATH=$PATH:/usr/sbin/rsct/bin:/opt/IBM/db2/V8.1/instance\
:/opt/IBM/db2/V8.1/ha/salinux
```

4. Source the /root/.bashrc file as follows:

```
(P)(S)(H) # source /root/.bashrc
```

5. Check that the environment variable CT\_MANAGEMENT\_SCOPE is indeed set to 2 by running the following command:

```
(P)(S)(H) # env | grep CT_MANAGEMENT_SCOPE
```

**Note:** if at any point during this topology setup you encounter an error message that says “CT\_MANAGEMENT\_SCOPE not set while configuring TSA”, explicitly set this environment variable by issuing this command:

```
# export CT_MANAGEMENT_SCOPE=2
```

## Prepare TSA cluster

Make sure that all TSA installations in your topology know about one another, and can communicate with one another in what is referred to as a TSA cluster domain. This is essential for management of HADR by TSA.

1. Run the following command as root to prepare the proper security environment between the TSA nodes:

```
(P)(S)(H) # preprnode hadr01 hadr02 hadr03
```

2. Issue the following command to create the cluster domain:

```
(P) # mkrpdomain hadr_domain hadr01 hadr02 hadr03
```

3. Now start the cluster domain as follows:

```
(P) # starttrpdomain hadr_domain
```

Note that all future TSA commands will be run relative to this active domain.

4. Ensure that hadr\_domain is online by issuing the following command:

```
(P) # lsrpdomain
```

You should see output similar to the following lines:

Name	OpState	RSCTActiveVersion	MixedVersions	TSPort
hadr_domain	Online	2.3.3.1	No	12347

5. Ensure that all nodes are online in the domain as follows:

```
(P) # lsrpnode
```

You should see output similar to the following lines:

Name	OpState	RSCTVersion
hadr01	Online	2.3.3.1
hadr02	Online	2.3.3.1
hadr03	Online	2.3.3.1

## Prepare nodes to host primary and standby instances

Create appropriate groups and users to manage DB2 on primary and standby nodes respectively. Remember to create passwords for all users created here:

1. As root, create the following groups for instance management:

```
(P)(S) # groupadd -g 999 db2iadm1
```

2. Create a group for fenced users:

```
(P)(S) # groupadd -g 998 db2fadm1
```

3. If you are planning to perform remote administration on DB2, then you must create a user group for the DAS:

```
(P)(S) # groupadd -g 997 db2asgrp
```

4. Add the following users for instance management:

```
(P) # useradd -g db2iadm1 -u 1005 -d /misc/homep/db2instp\  
-m db2instp
```

```
(S) # useradd -g db2iadm1 -u 1005 -d /misc/homes/db2insts\  
-m db2insts
```

5. Add the following fenced user:

```
(P)(S) #useradd -g db2fadm1 -u 1003 -d /misc/home/db2fenc1\  
-m db2fenc1
```

6. If you are planning to perform remote administration on DB2, then you must create DAS user:

```
(P)(S) # useradd -g db2asgrp -u 1002 -d /misc/home/db2as -m db2as
```

## Create primary and standby instances

Create the primary and standby instances (db2instp and db2insts) that will manage the HADR database (hadrdb).

**Note:** We create 32-bit instances in this example (`-w 32`). If you would like to create 64-bit instances instead, then change `-w 32` to `-w 64`. You must be running a 64-bit operating system to create a 64-bit instance.

1. As root, create the primary and standby database manager instances as follows:

```
(P) # db2icrt -w 32 -u db2fenc1 db2instp
```

```
(S) # db2icrt -w 32 -u db2fenc1 db2insts
```

**Attention:** Current TSA scripts shipped with DB2 *do not* allow primary and standby instances to share the same name. User modification of scripts will be necessary to use identical instances on both nodes.

2. If you plan to do remote administration of DB2, then create the DAS as follows:

```
(P)(S) # dasCRT -u db2as
```



3. Configure RSH to allow instance users to issue remote commands by adding the following to the file `/root/.rhosts`:

On (P), (S) and (H):

```
hadr01 db2instp
hadr02 db2insts
```

## Set up highly available IP addresses

Making an IP address highly available involves telling TSA of redundant network interface cards (NICs) (eth1) that you want to make equivalent to the primary NIC (eth0). Doing this allows TSA to restart an instance's failing IP address on one of the equivalent redundant NICs.

1. As root, find redundant NICs on the primary node (hadr01) that you would like to make equivalent to the primary NIC (eth0). Then, do the same on the standby node (hadr02):

```
(P) (S) # ifconfig -a | egrep 'eth|Mask'
```

You should see output similar to the following lines:

```
eth0      Link encap:Ethernet
          inet addr:9.26.96.84  Bcast:9.26.97.255
```

```
eth1      Link encap:Ethernet
          inet addr:192.168.1.1  Bcast:192.168.1.255
```

In this case, we choose to create an equivalency group with both NICs eth0 and eth1. Note that if there is only one NIC present on your own system, you will create an equivalency group of only this single adapter.

2. Choose any number of NICs on a node for the members of the NIC equivalency group. Note that for a NIC equivalency group with N members, N-1 NICs may fail and the IP address will be transparently moved to one of the surviving NICs in the group. Should the last (or only) NIC in the equivalency group fail, the node will be brought down (automatically by TSA) and any HADR resource groups will then be hosted at the surviving node.
3. Create an equivalency between the chosen NICs as follows (this first example assumes that eth0 and eth1 are active and equivalent on each cluster machine):

```
(P) # mkequ -D "(Name like 'eth0' | Name like 'eth1') &
NodeNameList='hadr01'" virpubnic_hadr01 IBM.NetworkInterface
```

```
(S) # mkequ -D "(Name like 'eth0' | Name like 'eth1') &
NodeNameList='hadr02'" virpubnic_hadr02 IBM.NetworkInterface
```

For machines with only eth0 active, create the equivalency relationships:

```
(P) # mkequ -D "Name like 'eth0' & NodeNameList='hadr01' "  
virpubnic_hadr01 IBM.NetworkInterface
```

```
(S) # mkequ -D "Name like ' ' & NodeNameList='hadr02' "  
virpubnic_hadr02 IBM.NetworkInterface
```

Note that for your own equivalency definitions, replace the actual primary node name for `hadr01` wherever it appears in the above examples, and replace the actual standby node name for `hadr02` wherever it appears in the above example.

For more details on creating the network equivalencies, consult the chapter on “Setting up a high available network” in the document *IBM Tivoli System Automation for Multiplatforms Base Component User's Guide*.

4. Validate that you have correctly created the equivalencies in step 3 above by issuing the following:

```
(P) # lsequ
```

You should see output similar to the following lines:

Displaying Equivalencies:

```
Virpubnic_hadr01  
Virpubnic_hadr02
```

Choose unused IP addresses for the primary and standby instances (for example, 9.26.96.90 and 9.26.96.91, respectively) that share the same net mask as the IP addresses of the base adapters. These will be the parameters passed to the registration script in the next section.

## Register instances with TSA for management

In this step, you register the instances so that TSA will be able to manage the instances, and later, HADR.

1. Register the instance (`db2instp` and `db2insts`) with TSA as follows:

```
(P) # regdb2salin -a db2instp -r -i 9.26.96.90  
(S) # regdb2salin -a db2insts -r -i 9.26.96.91
```

2. *Do not* issue `db2stop` or `db2start` after registering instances with TSA as resource groups. Instead, to stop the instance, use the following command:

```
# chrg -o offline <Resource_Group>
```

Where `<Resource_Group>` is the name given to the instance's resource group (`db2_db2instp_0-rg`).

3. Verify that the resource groups (db2\_db2instp\_0-rg and db2\_db2insts\_0-rg) were registered and are online by issuing the following command:

(P) # **getstatus**

You should see output similar to the following lines:

-- Resource Groups and Resources --

Group Name	Resources
-----	-----
db2_db2instp_0-rg	db2_db2instp_0-rs
db2_db2instp_0-rg	db2_db2instp_0-rs_ip
db2_db2insts_0-rg	db2_db2insts_0-rs
db2_db2insts_0-rg	db2_db2insts_0-rs_ip

-- Resources --

Resource Name	Node Name	State
-----	-----	-----
db2_db2instp_0-rs	hadr01	Online
db2_db2instp_0-rs_ip	hadr01	Online
-	-	-
db2_db2insts_0-rs	hadr02	Online
db2_db2insts_0-rs_ip	hadr02	Online

4. Validate that regdb2salin has not only registered the instances with TSA, but has also created the appropriate dependency relationships between the instances, their HA IP addresses, and the equivalencies that you created in step 2 above:

(P) # **lsrel**

You should see output similar to the following lines:

Displaying Managed Relations:

Name Class:Resource:Node[Source] ResourceGroup[Source]

db2\_db2instp\_0-rg\_IP\_do IBM.Application:db2\_db2instp\_0-rs  
db2\_db2instp\_0-rg

db2\_db2insts\_0-rg\_IP\_do IBM.Application:db2\_db2insts\_0-rs  
db2\_db2insts\_0-rg

### Create a DB2 HADR database

Now that you have created the primary and standby instances, (db2instp and db2insts respectively), you need to create a database (hadrdb) that you will make highly available through HADR.

1. As root, make sure that the database manager instance is started as follows. Remember, *do not* issue db2start as the instance is now controlled by TSA:

```
(P) # chrg -o online db2_db2instp_0-rg
(P) # getstatus
-- Resource Groups and Resources --
```

Group Name	Resources
-----	-----
db2_db2instp_0-rg	db2_db2instp_0-rs
db2_db2instp_0-rg	db2_db2instp_0-rs_ip
-	-
db2_db2insts_0-rg	db2_db2insts_0-rs
db2_db2insts_0-rg	db2_db2insts_0-rs_ip

-- Resources --

Resource Name	Node Name	State
-----	-----	-----
db2_db2instp_0-rs	hadr01	Online
-	-	-
db2_db2instp_0-rs_ip	hadr01	Online
-	-	-
db2_db2insts_0-rs	hadr02	Online
-	-	-
db2_db2insts_0-rs_ip	hadr02	Online

If you would like to better understand how HADR works, please view the following Flash demo at:

[http://demos.dfw.ibm.com/on\\_demand/Demo/IBM\\_Demo\\_DB2\\_HADR-Jan05.html?S=index&S=DC](http://demos.dfw.ibm.com/on_demand/Demo/IBM_Demo_DB2_HADR-Jan05.html?S=index&S=DC)

2. As primary instance owner (db2instp), create the database (hadrdb) that you will later make highly available with HADR as follows.

```
(P) % db2 create database hadrdb on <database_directory_path>
```

Where <database\_directory\_path> is any valid path on the primary node, for example /misc/homep/db2instp.

**Attention:** Since the DB2 TSA scripts for HADR require distinct names for the primary and standby instances, we recommend that absolute container path names do not contain the instance name as part of the path, as that increases the possibilities of error in managing database storage in a HADR environment.

For example, use `ALTER TABLESPACE TSPACE1 ADD (FILE '/data/hadrdb/tspace1/cont2' 2000)` where the path `/data/hadrdb/tspace1` must exist on both primary and standby nodes. Avoid using `ALTER TABLESPACE TSPACE1 ADD (FILE '/misc/homep/db2instp/cont2' 2000)` where the path `/misc/homep/db2instp` must exist on both primary and standby. Relative path names can be used as usual, for example, `ALTER TABLESPACE TSPACE1 ADD (FILE 'cont2' 2000)`.

3. For HADR to work, you must change the database's (hadrdb) default circular logging to archive logging by issuing the following command:

```
(P) % db2 update db cfg for hadrdb using LOGRETAIN ON
```

4. Now we must create a backup copy of the primary database (hadrdb) that will later be restored on the standby instance, and act as the standby database of the HADR pair. The backup image will be written to the instance owner's home directory, for example `/misc/homep/db2instp`:

```
(P) % db2 backup database hadrdb
```

5. Transfer the backup image of the primary database (hadrdb) to the standby instance owner's home directory (for example, `/misc/homes/db2insts`). Here is an example that uses secure copy (scp):

```
(P) % scp /misc/homep/db2instp/<backup_image_name> \
db2insts@hadr02:/misc/homes/db2insts
```

Where `<backup_image_name>` is the name of the backup image file.

6. As standby instance owner (db2insts), create the standby database on the standby node (hadr02) as follows:

```
(S) % db2 restore database hadrdb replace history file
```

7. Allow TCPIP communication to both the primary and standby instance (for example, db2instp and db2insts respectively). Note that `DB2_db2instp 60000/tcp` should be in the `/etc/services` file on the primary node (hadr01), and `DB2_db2insts 60000/tcp` should be in the `/etc/services` file on the standby node (hadr02) before issuing the following commands:

```
(P)(S) % db2set DB2COMM=tcPIP
```

```
(P) % db2 update dbm cfg using SVCENAME DB2_db2instp
```

```
(S) % db2 update dbm cfg using SVCENAME DB2_db2insts
```

8. DB2 must start an HADR service on the primary node (hadr01), to ship logs to the standby node (hadr02). Thus, as root, we must add the following to the /etc/services file on the primary node:

```
(P) # hadrinstp 18819/tcp
```

9. DB2 must start an HADR service on the standby node (hadr02), to accept logs from the primary node (hadr01). Thus, as root, add the following to the /etc/services file on the standby node:

```
(S) # hadrinsts 18820/tcp
```

10. As primary instance owner (db2instp), enable HADR on the primary database (hadrdb) as follows:

```
(P) % db2 update db cfg for hadrdb using \ HADR_LOCAL_HOST hadr01
(P) % db2 update db cfg for hadrdb using \ HADR_REMOTE_HOST hadr02
(P) % db2 update db cfg for hadrdb using \ HADR_LOCAL_SVC 18819
(P) % db2 update db cfg for hadrdb using \ HADR_REMOTE_SVC 18820
(P) % db2 update db cfg for hadrdb using \ HADR_REMOTE_INST db2insts
```

11. As standby instance owner (db2insts), enable HADR on the standby database (hadrdb) as follows:

```
(S) % db2 update db cfg for hadrdb using \ HADR_LOCAL_HOST hadr02
(S) % db2 update db cfg for hadrdb using \ HADR_REMOTE_HOST hadr01
(S) % db2 update db cfg for hadrdb using \ HADR_LOCAL_SVC 18820
(S) % db2 update db cfg for hadrdb using \ HADR_REMOTE_SVC 18819
(S) % db2 update db cfg for hadrdb using \ HADR_REMOTE_INST db2instp
```

12. Choose a synchronization mode to run the HADR pair in. There are three modes: synchronous, near-synchronous, and asynchronous. Synchronous mode provides the greatest protection against transaction loss, but has the slowest transaction time. Asynchronous mode provides the least protection against transaction loss, but has the fastest transaction time. Near-synchronous mode provides more protection than asynchronous mode and its transaction time is a little faster than that of synchronous mode. We use the synchronous mode in this case:

```
(P)(S) % db2 update db cfg for hadrdb using HADR_SYNCMODE\ sync
```

For more information on the various HADR synchronization modes, see the section in DB2 Information Center at the Web site:

<http://publib.boulder.ibm.com/infocenter/db2help/index.jsp?topic=co.ibm.db2.udb.doc/admin/r0011445.htm>

13. Specify the amount of time (in seconds) that the HADR process waits before considering a communication attempt to have failed:

```
(P)(S) db2 update db cfg for hadrdb using HADR_TIMEOUT 30
```

**Note:** Only modify this parameter if you are certain that performance will be improved as a result.

14. As instance owner, verify that you have set the database configuration parameters correctly in 10 and 11 above:  

```
(P) (S) % db2 get db cfg for hadrdb
```
15. As primary instance owner (db2instp), update the alternate server for the primary database (hadrdb) as follows:  

```
(P) % db2 update alternate server for database hadrdb using hostname <instance_hostname> port 60000
```

Where <instance\_hostname> is the IP address used in the registration of the standby DB2 instance (in our example, 9.26.96.91).
16. As standby instance owner (db2insts), update the alternate server for the standby database (hadrdb) as follows:  

```
(S) % db2 update alternate server for database hadrdb using hostname <instance_hostname> port 60000
```

Where <instance\_hostname> is the IP address used in the registration of the primary DB2 instance (in our example, 9.26.96.90).
17. As standby instance owner (e.g., db2insts), start HADR on the standby node (for example, hadr02) as follows:  

```
(S) % db2 start hadr on db hadrdb as standby
```

Ignore the message: “Logindex build was not enabled before HADR was started”.

Make sure that you keep the following points in mind:

  - a. Always start HADR on the standby *before* starting HADR on the primary.
  - b. At each client, ensure that the primary node (hadr01) is cataloged. Here is an example:  

```
(H) % db2 catalog tcpip node db2instp remote hadr01 server 60000
```
  - c. Each client must connect to the primary instance at least once, so as to pickup alternate server information. Also, clients *must* connect through a TCPIP node entry, *not* through a local node entry. If you wish to connect through a local node entry (for example, the client resides on the same node as the database), then you must create a TCPIP catalog entry that references the local instance (that is, a loop back entry)
  - d. If using Java Database Connectivity (JDBC) type 4 clients, client reroute information will not be picked up from the local db directory. Applications

have to be designed to store and recover the alternate server information in a text file.

18. As primary instance owner (db2instp), start HADR on the primary node (hadr02) as follows:

```
(P) % db2 start hadr on db hadrdb as primary
```

Ignore the message: "Logindex build was not enabled before HADR was started".

19. As instance owner (db2instp), ensure that the HADR pair is in Peer state as follows:

```
(P) (S) % db2 get snapshot for db on hadrdb
```

You should see output similar to this on the primary node (hadr01):

```
HADR Status Role = Primary
State = Peer
Synchronization mode = sync
Connection status = Connected, 11/08/2006 13:50:19.620630
Heartbeats missed = 0
Local host = hadr01
Local service = 18819
Remote host = hadr02
Remote service = 18820
Remote instance = db2insts
timeout(seconds) = 30
Primary log position(file, page, LSN) = S0000000.LOG, 0, 00000000007D0000
Standby log position(file, page, LSN) = S0000000.LOG, 0, 00000000007D0000
```

You should see output similar to this on the standby node (hadr02):

```
HADR Status Role = Standby
State = Peer
Synchronization mode = sync
Connection status = Connected, 11/08/2006 13:50:19.620630
Heartbeats missed = 0
Local host = hadr02
Local service = 18820
Remote host = hadr01
Remote service = 18819
Remote instance = db2instp
timeout(seconds) = 30
Primary log position(file, page, LSN) = S0000000.LOG, 0, 00000000007D0000
Standby log position(file, page, LSN) = S0000000.LOG, 0, 00000000007D0000
```



## Register HADR with TSA for automatic management

In this step, we enable TSA to automatically monitor and manage the HADR pair. We do this by registering the HADR pair as a resource group with TSA.

*Do not* manually issue DB2 “Takeover” commands after registering HADR as a resource group with TSA. Instead, use the following command to manually restart HADR:

```
# chrg -o online <resource_group_name>
```

Where <resource\_group\_name> is the name TSA gives to HADR resource group, for example db2hadr\_hadrdb-rg.

For operator controlled failover, issue the following command only:

```
# rgreq -o move -n <node> <resource_group_name>
```

Where <node> is the node (hadr01) at which the database (hadrdb) has “Primary” HADR role. <resource\_group\_name> is the name TSA gives to HADR resource group, for example db2hadr\_hadrdb-rg.

Now follow these steps:

1. As root on the primary node (hadr01), create a resource group for the HADR pair as follows:

```
(P) # reghadrsalin -a db2instp -b db2insts -d hadrdb
```

2. Check the status of all TSA resource groups by issuing this command:

```
(P) # getstatus
```

You should see output similar to the following lines if you set up an HA IP address (as shown in “Set up highly available IP addresses” on page 299). If you did not set up an HA IP address, then the output below will not contain any IP resources:

```
-- Resource Groups and Resources --
```

Group Name	Resources
-----	-----
db2_db2instp_0-rg	db2_db2instp_0-rs
db2_db2instp_0-rg	db2_db2instp_0-rs_ip
-	-
db2_db2insts_0-rg	db2_db2insts_0-rs
db2_db2insts_0-rg	db2_db2insts_0-rs_ip

```
-- Resources --
```

Resource Name	Node Name	State
-----	-----	-----

db2_db2instp_0-rs	hadr01	Online
-	-	-
db2_db2instp_0-rs_ip	hadr01	Online
-	-	-
db2_db2insts_0-rs	hadr02	Online
-	-	-
db2_db2insts_0-rs_ip	hadr02	Online
-	-	-
db2hadr_hadrdb-rs	hadr01	Online
db2hadr_hadrdb-rs	hadr02	Offline
-	-	-

**Note:** The node at which the HADR resource is online, for example, db2hadr\_hadrdb-rs, is the node at which the database is in primary mode.

### 10.3.3 Testing topology response to common failures

For all following test cases, we assume that hadrdb is primary on hadr01, and all of the instance resource groups are online.

Next we explain the meaning of the states that you see after issuing the **getstatus** command:

- ▶ For instance resources:
  - Online: Indicates instance is up.
  - Offline: Indicates instance is down.
  - Unknown: Indicates the state of the instance is unknown.
- ▶ For HADR resources:
  - Online: Indicates HADR database is primary at node name.
  - Offline: Indicates HADR database is standby at the node name.
  - Unknown: Indicates HADR database is not in Peer state OR HADR database is down.

#### Controlled failover testing

These are the steps for conducting controlled failover testing:

1. As root, move “db2hadr\_hadrdb-rs” resource from the primary node (hadr01) to the standby node (e.g., hadr02); in other words, perform a controlled failover as follows (ignore the “token” message):

```
(P) # rgreq -o move -n hadr01 db2hadr_hadrdb-rg
```

Ignore the “token” message.

As root, check that the primary database (hadrdb) has successfully failed over to the standby node (hadr02) as follows:

(P) # **getstatus**

You should see output similar to the following lines if you set up an HA IP address. If you did not set up HA IP addresses, then the output below will not include any IP resources:

```
-- Resource Groups and Resources --
      Group Name          Resources
      -----
db2_db2instp_0-rg  db2_db2instp_0-rs
db2_db2instp_0-rg  db2_db2instp_0-rs_ip
      -
db2_db2insts_0-rg  db2_db2insts_0-rs
db2_db2insts_0-rg  db2_db2insts_0-rs_ip

-- Resources --
      Resource Name          Node Name    State
      -----
db2_db2instp_0-rs          hadr01      Online
      -
db2_db2instp_0-rs_ip      hadr01      Online
      -
db2_db2insts_0-rs          hadr02      Online
      -
db2_db2insts_0-rs_ip      hadr02      Online
      -
db2hadr_hadrdb-rs          hadr01      Online
db2hadr_hadrdb-rs          hadr02      Offline
      -
```

2. Return hadrdb back to being primary on hadr01 as follows:

(P) # **rgreq -o move -n hadr02 db2hadr\_hadrdb-rg**

Ignore the “token” message.

## Testing instance failure

Instance failure test should be performed on both the primary and standby instances.

### **Primary instance**

Use the following steps to test the primary instance (db2instp) failed situation:

1. As primary instance owner, simulate a primary instance failure as follows:

(P) % **db2\_kill**

2. On the standby node (hadr02), issue the following command repeatedly until you see output similar to what you saw the first time you ran the command, that is: db2\_db2instp\_0-rs is Online again):

(S) # **getstatus** Please

Note: It may take up to 2 minutes to see the instance restarted automatically.

### ***Standby instance***

Use the following steps to test the standby instance failed situation:

1. As standby instance owner, simulate a standby instance failure as follows:

(S) # **db2\_kill**

2. On the primary node (hadr01), issue the following repeatedly until you see output similar to what you saw the first time you ran the command, that is: db2\_db2insts\_0-rs is Online again:

(P) # **getstatus**

Note: It may take up to 2 minutes to see the instance restarted.

### **Testing resource group failure**

Resource group failure testing should be performed on both the primary and the standby instance resource groups.

#### ***Primary instance resource group***

Use the following steps to test the primary instance resource group failed situation:

1. As root, bring the primary instance resource group (db2\_db2instp\_0-rg) offline as follows:

(P) # **chrg -o offline** db2\_db2instp\_0-rg

2. Issue the following command and observe that the primary instance resource group goes offline:

(P) # **getstatus**

3. Now bring the primary instance resource group back online by issuing the following command:

(P) # **chrg -o online** db2\_db2instp\_0-rg

4. As root, verify that the primary instance resource group is back online as follows:

(P) # **getstatus**

## Standby instance resource group

Follow these steps to test the standby instance resource group failed situation:

1. As root, bring the standby instance resource group (db2\_db2insts\_0-rg) offline as follows:  
(S) # **chrg -o offline** db2\_db2insts\_0-rg
2. Issue the following command and observe that the standby instance resource group goes offline:  
(S) # **getstatus**
3. As root, bring the standby instance resource group back online as follows:  
(S) # **chrg -o online** db2\_db2insts\_0-rg
4. As root, verify that the standby instance resource group has been restarted successfully:  
(S) # **getstatus**

## Testing network adapter failure

To test the adapter failure (eth0), perform the following steps:

1. Pull the cable on the NIC currently hosting the primary instance's IP address (eth0). You should see a brief interruption in service while the IP address fails over to another adapter in the equivalency group (eth1).

Note that should the cable from the only Ethernet adapter on the box, or the cable to the last surviving Ethernet adapter on the box be pulled, the system behavior will be identical to that described in "Node failure" on page 312".

2. Check that the topology has returned to its original state before the cable was pulled:

(S) # **getstatus**

You should see output similar to this:

```
-- Resource Groups and Resources --
      Group Name      Resources
      -----
db2_db2instp_0-rg    db2_db2instp_0-rs
db2_db2instp_0-rg    db2_db2instp_0-rs_ip
                    -
db2_db2insts_0-rg    db2_db2insts_0-rs
db2_db2insts_0-rg    db2_db2insts_0-rs_ip

-- Resources --
      Resource Name      Node Name      State
```

-----	-----	-----
db2_db2instp_0-rs	hadr01	Online
-	-	-
db2_db2instp_0-rs_ip	hadr01	Online
-	-	-
db2_db2insts_0-rs	hadr02	Online
-	-	-
db2_db2insts_0-rs_ip	hadr02	Online
-	-	-
db2hadr_hadrdb-rs	hadr01	Online
db2hadr_hadrdb-rs	hadr02	Offline

### Node failure

For this test case to work, you must *uncomment* the line in the example below in */opt/IBM/db2/V8.1/ha/salinux/hadr\_start.ksh* on both the primary and standby nodes. You do this in order to allow TAKEOVER BY FORCE if the HADR pair drops out of Peer state before TSA can issue a failover to the standby database.

If HADR is not operating in synchronization mode SYNC, uncommenting this line could cause the standby database to takeover as primary at a time when it is not in SYNC with the primary database that failed. If this is the case, then later reintegration of the HADR pair might fail, and the standby database might have to be re-established using a backup image of the current primary database.

On (P) and (S), uncomment (remove the comment mark) in */opt/IBM/db2/V8.1/ha/salinux/hadr\_start.ksh*:

```
#su - ${instance_to_start?} -c "db2 takeover hadr on db\
${DB2HADRDBNAME? } by force >> $temp_log_file" > /dev/null
```

1. Check that the status of all resources are normal as follows:

(S) # **getstatus**

Output similar to the following coding should be seen if you set up HA IP addresses. If you did not set up HA IP addresses, then the following output will not contain any IP resources:

```
-- Resource Groups and Resources --
      Group Name          Resources
      -----
db2_db2instp_0-rg      db2_db2instp_0-rs
db2_db2instp_0-rg      db2_db2instp_0-rs_ip
-
db2_db2insts_0-rg      db2_db2insts_0-rs
db2_db2insts_0-rg      db2_db2insts_0-rs_ip

-- Resources --
```

Resource Name	Node Name	State
-----	-----	-----
db2_db2instp_0-rs	hadr01	Online
-	-	-
db2_db2instp_0-rs_ip	hadr01	Online
-	-	-
db2_db2insts_0-rs	hadr02	Online
-	-	-
db2_db2insts_0-rs_ip	hadr02	Online
-	-	-
db2hadr_hadrdb-rs	hadr01	Online
db2hadr_hadrdb-rs	hadr02	Offline

- As root, simulate a failure of the primary node (hadr01) by rebooting Linux:

(P) # **init 6**

- As root, check status of recovery by issuing the following command repeatedly:

(S) # **getstatus**

After five or more minutes, output similar to the following coding should be seen if you set up an HA IP address. If you did not set up an HA IP address, then no IP resources will be seen in the following output:

```
-- Resource Groups and Resources --
Group Name          Resources
-----
db2_db2instp_0-rg   db2_db2instp_0-rs
db2_db2instp_0-rg   db2_db2instp_0-rs_ip
-                   -
db2_db2insts_0-rg    db2_db2insts_0-rs
db2_db2insts_0-rg    db2_db2insts_0-rs_ip

-- Resources --
Resource Name      Node Name  State
-----
db2_db2instp_0-rs  hadr01    Online
-                 -
db2_db2instp_0-rs_ip hadr01    Online
-                 -
db2_db2insts_0-rs  hadr02    Online
-                 -
db2_db2insts_0-rs_ip hadr02    Online
-                 -
db2hadr_hadrdb-rs  hadr01    Unknow
db2hadr_hadrdb-rs  hadr02    Unknow
```

Verify that the HADR database is now primary on node hadr02 using the DB2 GET SNAPSHOT command:

```
% db2 get snapshot for db on hadrdb
```

4. Once the old primary machine (hadr01) comes back online, you can re-establish the HADR pair. As the original primary instance owner, db2instp, run the following command:

```
(P) % db2 start hadr on db hadrdb as standby
```

5. The HADR pair should now be re-established. You can issue a **getstatus** to check if you like. To bring the primary database back to hadr01, issue the following (ignore the “token” message):

```
(P) # rgreq -o move -n hadr02 db2hadr_hadrdb-rg
```

6. 7. Check that HADR has returned to its original state before the primary node failure:

```
(P) # getstatus
```

Output similar to the following lines should be seen:

```
-- Resource Groups and Resources --
      Group Name          Resources
      -----
db2_db2instp_0-rg      db2_db2instp_0-rs
db2_db2instp_0-rg      db2_db2instp_0-rs_ip
-                      -
db2_db2insts_0-rg      db2_db2insts_0-rs
db2_db2insts_0-rg      db2_db2insts_0-rs_ip

-- Resources --
      Resource Name          Node Name    State
      -----
db2_db2instp_0-rs          hadr01      Online
-                          -
db2_db2instp_0-rs_ip      hadr01      Online
-                          -
db2_db2insts_0-rs          hadr02      Online
-                          -
db2_db2insts_0-rs_ip      hadr02      Online
-                          -
db2hadr_hadrdb-rs          hadr01      Online
db2hadr_hadrdb-rs          hadr02      Offline
```





# Q replication

In this chapter we provide an introduction to Q replication and a simple overview of its structure. We also give you step-by-step instructions for setting up a unidirectional Q replication.

## 11.1 Introduction

Q replication is a replication solution that uses WebSphere MQ message queues to transmit transactions between source and target databases.

Q replication is divided into three types:

- Unidirectional:

In *unidirectional Q replication*, changes are captured at the source and replicated to the target. Unidirectional Q replication can happen from one source to one target or many targets. In unidirectional replication, typically the target is used for read-only.

- Bidirectional:

In *bidirectional Q replication*, two tables on two servers replicate each other. Changes made on either table are replicated to the corresponding table. You cannot replicate a subset of rolls, and each table must have the same number of columns and data types for those columns, though you can have different schema and table names.

- Peer-to-peer:

In *peer-to-peer Q replication*, tables are replicated between two or more servers. As in bidirectional Q replication, the replicated tables must all be of the same structure, though each table can have its own schema and table name.

Q replication can be set up in many different configurations. You can use Q replication for replication between databases on the same server or remote servers. You can set up a one-to-many relationship or many-to-one relationship.

For more details on different Q replication configurations, refer to the DB2 Information Center:

<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

Next we describe the basic structure of Q replication:

- A Q Capture program that runs on the source server reads the DB2 recovery log for changes to the source tables specified for replication. It then places the messages in a queue called *send queue*. The Q Capture program uses a set of DB2 tables called *Q Capture Control tables*. The capture control tables store information about the replication sources, corresponding targets, MQ queues being used, and other data.

- Q subscriptions define the source and target tables to be replicated. Q subscriptions need a replication queue map to identify the WebSphere MQ queues used for sending and receiving transactions.
- The Q Apply program runs on the target server, receiving the messages and applying the transactions to the target tables. The Q Apply program also has a set of DB2 tables called *Q Apply Control tables*.

In the rest of this chapter we focus on a unidirectional Q replication setup.

## 11.2 Unidirectional setup

This section provides a step-by-step example of a unidirectional Q replication setup between two remote servers. See Figure 11-1.

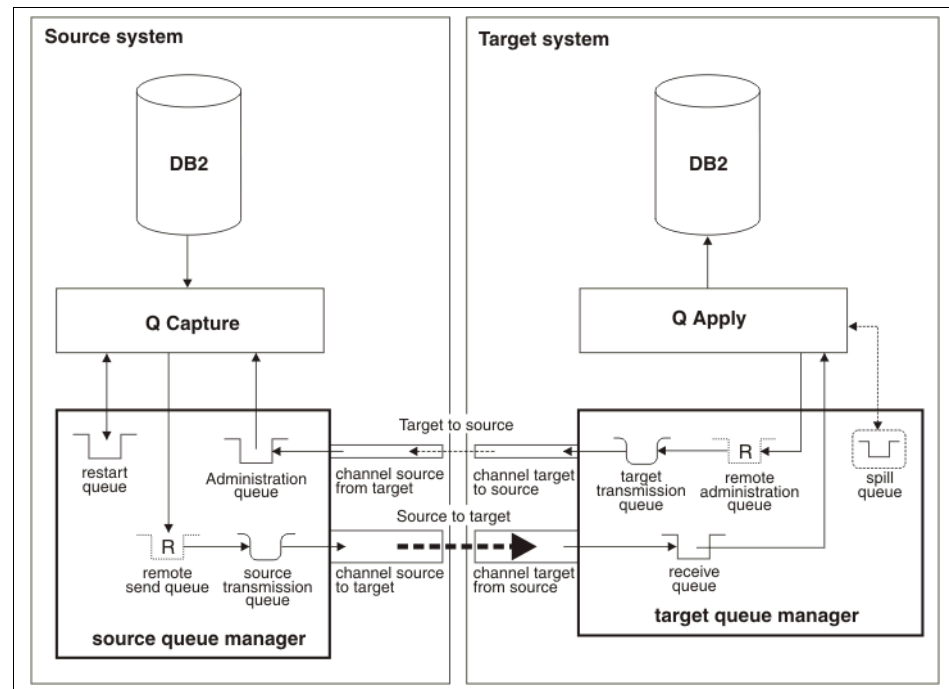


Figure 11-1 Unidirectional setup diagram

The following tasks must be completed for setting up a unidirection Q replication:

1. Set up the database on both the source and target database servers.
2. Set up WebSphere MQ objects on the source server and the target server.
3. Start the listener and the channel on the source and the target.
4. Optional: Test the queues.
5. Create the Q Capture Control tables.
6. Create the Q Apply Control tables.
7. Create a Q subscription.
8. Configure target to source server connectivity.
9. Start Q Capture.
10. Start Q Apply.

In our example, we used two Windows servers running DB2 Version 9.1 and WebSphere MQ 6.0. The step-by-step process to set up a unidirectional Q replication with remote servers is as follows:

1. Initial database setup on the source and the target server:
  - a. The source database (SAMPLE in our test case) must have the database configuration parameter LOGRETAIN set to RECOVERY.  
**db2 update database configuration for *sample* using LOGRETAIN RECOVERY**  
  
A backup will be required if you are setting the LOGRETAIN to recovery for the first time.
  - b. Identify your target database. For our example, the target database is TGTDB.
2. Setup of WebSphere MQ objects on the source server:
  - a. Create a queue manager QMGR1 from the command window:  
**crtmqm QMGR1**
  - b. Start the Q manager that you have just created:  
**strmqm QMGR1**
  - c. To create the needed MQ objects, run the following command with the example script shown in Example 11-1. Remember to replace the IP and port number with your required IP and port number:  
**runmqsc QMGR1 < QMGR1.DEF**

### Example 11-1 QMGR1.DEF

---

**\*\*This script creates the following objects within queue manager QMGR1  
\*\*at the source:**

**\*\*A local administration queue**

DEF QLOCAL('SAMPLE\_ADMINQ') DESCR('Capture Admin Queue') REPLACE

**\*\*A local restart queue**

DEF QLOCAL('SAMPLE\_RESTARTQ') DESCR('Capture Restart Queue') REPLACE

**\*\*A remote queue definition for Q Capture to put data messages**

DEF QREMOTE('SAMPLE2TGTDQB') XMITQ('QMGR2') RNAME('SAMPLE2TGTDQB')  
RQMNAME('QMGR2') REPLACE

**\*\*A local transmission queue**

DEF QLOCAL('QMGR2') USAGE(XMITQ) REPLACE

**\*\*A sender channel from QMGR1 to queue manager QMGR2 using transmission  
\*\*queue**

**\*\*Replace the ip with your ip and the port number with your port number**

DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(SDR) TRPTYPE(TCP)  
CONNNAME('9.43.86.78 (1416)') XMITQ('QMGR2') REPLACE

**\*\*A receiver channel from QMGR2**

DEF CHANNEL('QMGR2.TO.QMGR1') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE

---

### 3. Setup of WebSphere MQ objects on the target server:

- a. Create a queue manager with a different name as the source from the command window:

**crtmqm QMGR2**

- b. Start the Q manager that you have just created:

**strmqm QMGR2**

- c. To create the needed MQ objects, run the following command with the example script shown in Example 11-2. Remember to replace the IP and port number with your required IP and port number:

**runmqsc QMGR2 < QMGR2.DEF**

### *Example 11-2 QMGR2.DEF*

---

**\*\*This script creates the following objects within queue manager QMGR2  
\*\*at the target:**

**\*\*A remote queue definition that points to the administration queue at  
\*\*the source**

```
DEF QREMOTE('SAMPLE_ADMINQ') XMITQ('QMGR1') +  
RNAME('SAMPLE_ADMINQ') RQMNAME('QMGR1') REPLACE
```

**\*\*A model queue definition for spill queues that hold data messages  
\*\*from Q Capture during the load**

```
DEF QMODEL('IBMQREP.SPILL.MODELQ') DEFSOPT(SHARED) +  
MAXDEPTH(500000) MSGDLVSQ(FIFO) DEFTYPE(PERMDYN) REPLACE
```

**\*\*A local queue for Q Apply to get data messages from the Q Capture  
\*\*program at the source**

```
DEF QLOCAL('SAMPLE2TGTDQB') REPLACE
```

**\*\*A local transmission queue**

```
DEF QLOCAL('QMGR1') USAGE(XMITQ) REPLACE
```

**\*\*A sender channel from QM2 to queue manager QMGR1 using transmission  
\*\*queue QMGR1**

**\*\*Replace the ip with your ip and the port number with your port number**

```
DEF CHANNEL('QMGR2.TO.QMGR1') CHLTYPE(SDR) TRPTYPE(TCP) +  
CONNAME('9.43.86.75 (1415)') XMITQ('QMGR1') REPLACE
```

**\*\*A receiver channel from QMGR1**

```
DEF CHANNEL('QMGR1.TO.QMGR2') CHLTYPE(RCVR) TRPTYPE(TCP) REPLACE
```

---

#### **4. Start both the listener and channel on the source and the target:**

- a. On the source, from a command window start the listener:

```
RUNMQLSR -T TCP -M QMGR1 -P 1415
```

- b. On the target, from a command window, start the listener:

```
RUNMQLSR -T TCP -M QMGR2 -P 1416
```

- c. On the source, from a command window, start the channel:

To start the channel, first use the RUNMQSC command to start the interactive MQSC command line:

```
RUNMQSC QMGR1
```

This command starts the channel:

```
START CHANNEL(QMGR1.TO.QMGR2)
```

This command ends the **runmqsc** session.

**END**

- d. On the target from a command window start the channel:

```
RUNMQSC QMGR2  
START CHANNEL(QMGR2.TO.QMGR1)  
END
```

5. Optional: Test the queues:

We test the message queue setup by putting a message from the source system and try to get the message from the target system.

- a. In the source, we put a message “Test message to send” into the message queue as shown in Example 11-3.
- i. Press Enter to send the message.
  - ii. Press Enter to end the prompt.

*Example 11-3 amqsput*

---

```
>amqsput SAMPLE2TGTDDBQ QMGR1  
Sample AMQSPUTO start  
target queue is SAMPLE2TGTDDBQ  
Test message to send  
Sample AMQSPUTO end
```

---

- b. Example 11-4 shows how to receive the message from the target. Press Enter to end the prompt.

*Example 11-4 amqsget*

---

```
>amqsget SAMPLE2TGTDDBQ QMGR2  
Sample AMQSGETO start  
message <Test message to send>  
no more messages  
Sample AMQSGETO end
```

---

6. Creating the Q Capture Control tables:
  - a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box. See Figure 11-2.

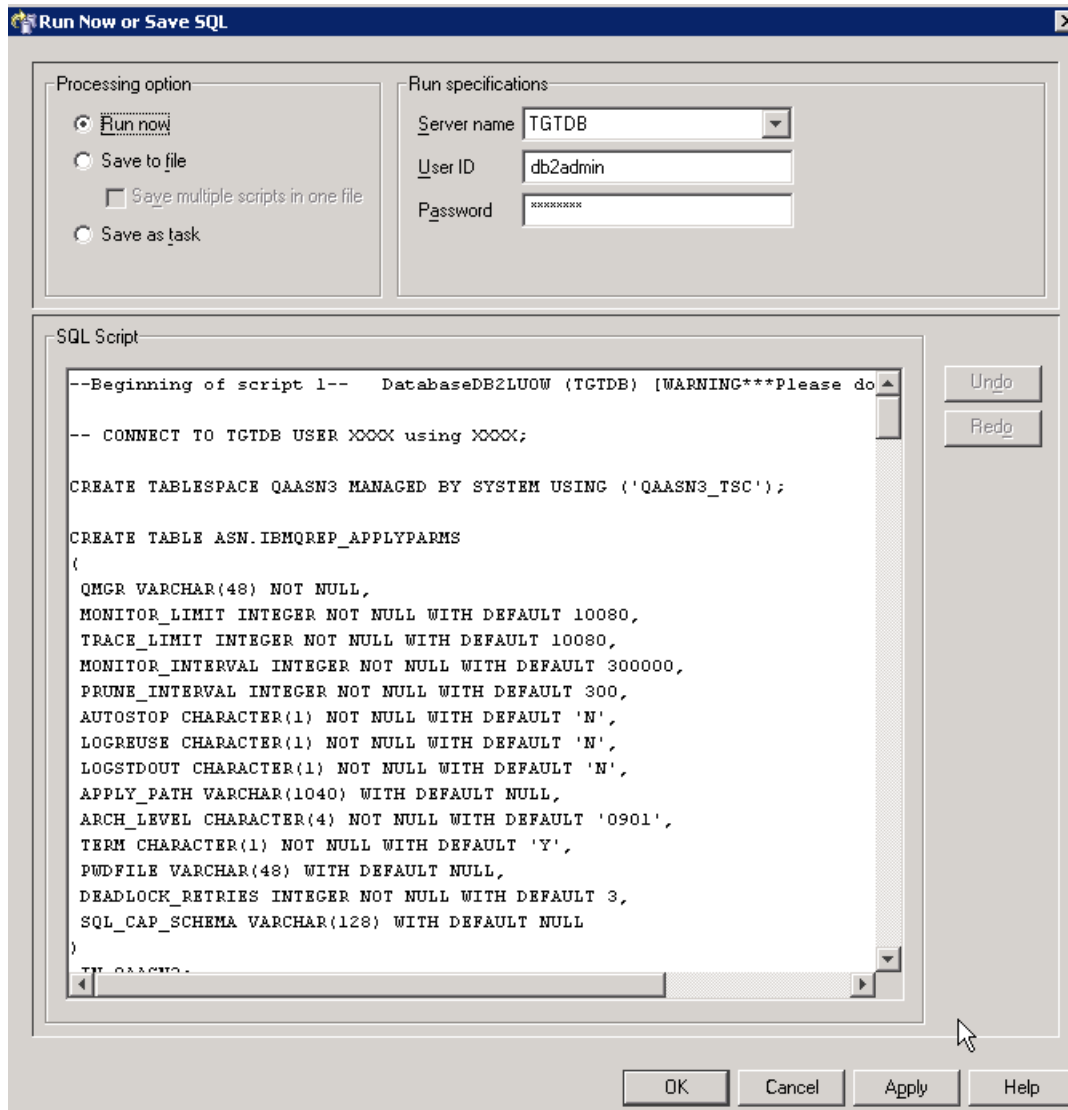


Figure 11-2 Replication Center Launchpad



- b. Click **1. Create Q Capture Control Tables** from the Replication Center Launchpad. This opens the Create Q Capture Control Tables Wizard. From the first panel in the wizard, choose **Typical** or **Custom**, then click **Next**. See Figure 11-3.

**Note:** When creating the Q Capture Control tables, choosing the Custom option allows you to define the tablespace or tablespaces you want the tables to reside in.

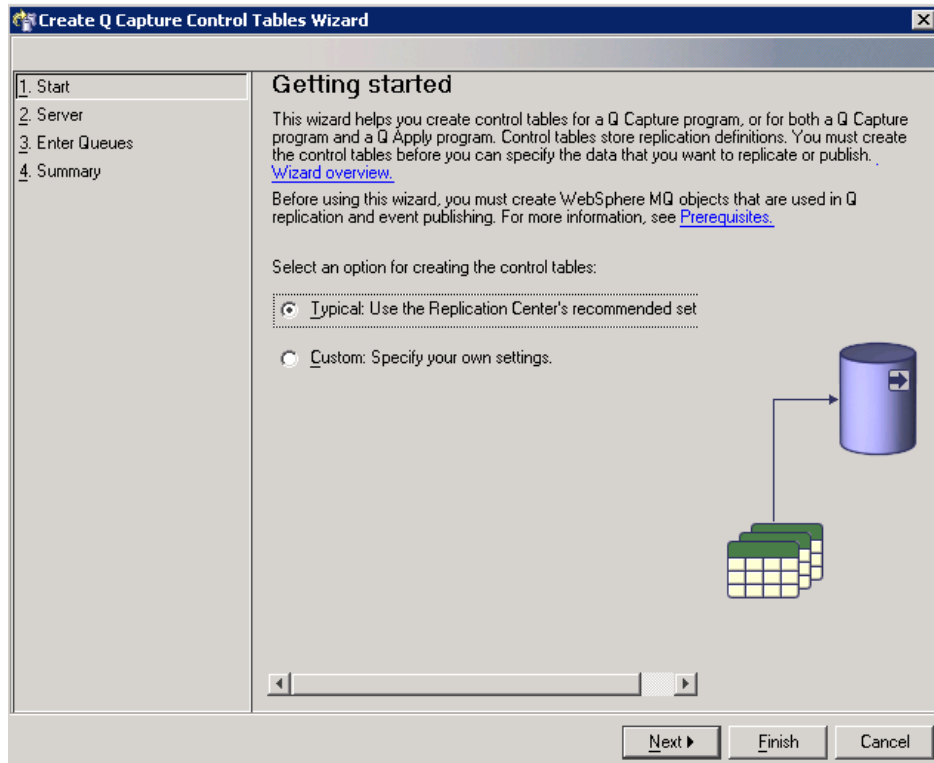


Figure 11-3 Create Q Capture Control Tables Wizard, panel 1

- c. Select the Q Capture server from the list (source server). This should populate the User ID, Password, and Q Capture schema. The default schema is ASN, but can be changed if necessary. Click **Next** to proceed. See Figure 11-4.

**Create Q Capture Control Tables Wizard**

1. Start  
2. Server  
3. Enter Queues  
4. Summary

**Specify a Q Capture server and a Q Capture schema**

Specify the Q Capture server. The Q Capture server is the DB2 database (Linux, UNIX, Windows) or DB2 subsystem (z/OS) that contains your source data. The control tables will be created on this server. Next, specify a schema to identify the Q Capture program and its unique set of control tables.

Q Capture server:

User ID:

Password:

Q Capture schema:

DB2 subsystem:

Database:

☐ Create both Q Capture and Q Apply control tables on this server  
[When do you want to choose this option?](#)

Back Next Finish Cancel

Figure 11-4 Create Q Capture Control Tables Wizard, panel 2

- d. Enter the Queue manager name for the source server. In our example, the Queue manager is QMGR1. Enter the Administration queue and Restart queue, which are SAMPLE\_ADMINQ and SAMPLE\_RESTARTQ respectively in our example. Click **Next** to proceed to the summary panel. See Figure 11-5.

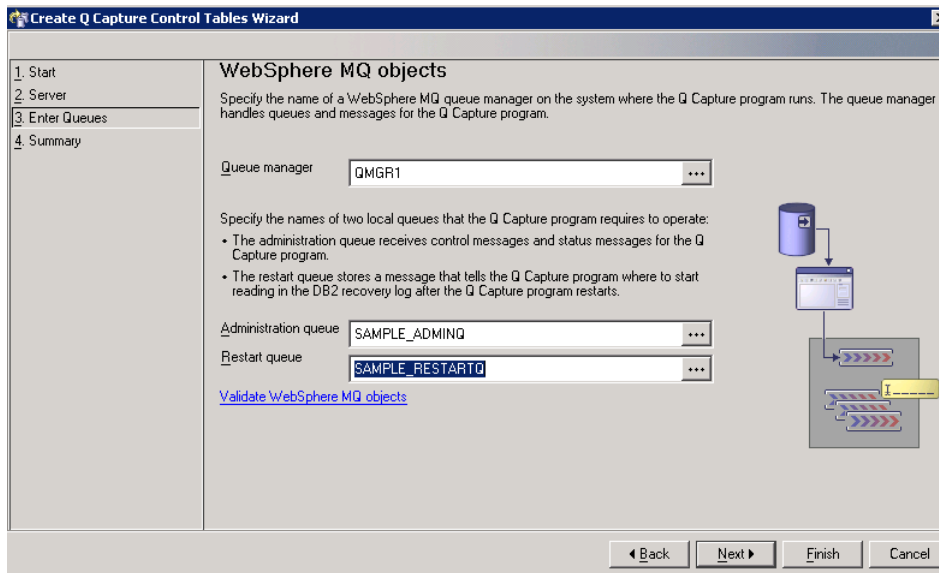


Figure 11-5 Create Q Capture Control Tables Wizard, panel 3

e. Review the Summary panel and click **Finish**. See Figure 11-6.

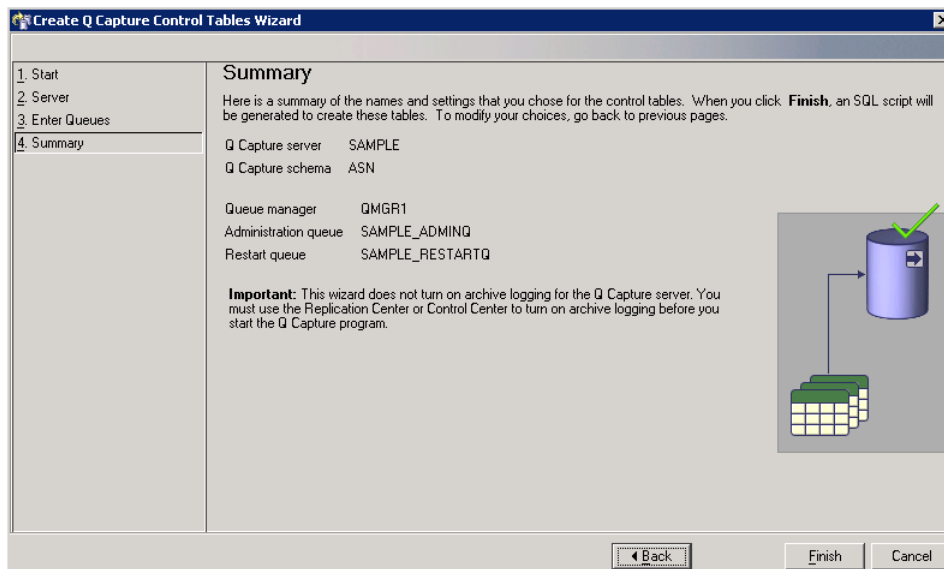


Figure 11-6 Create Q Capture Control Tables Wizard: Summary

- f. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 11-7. After saving the SQL select **Run now** and click **OK**.

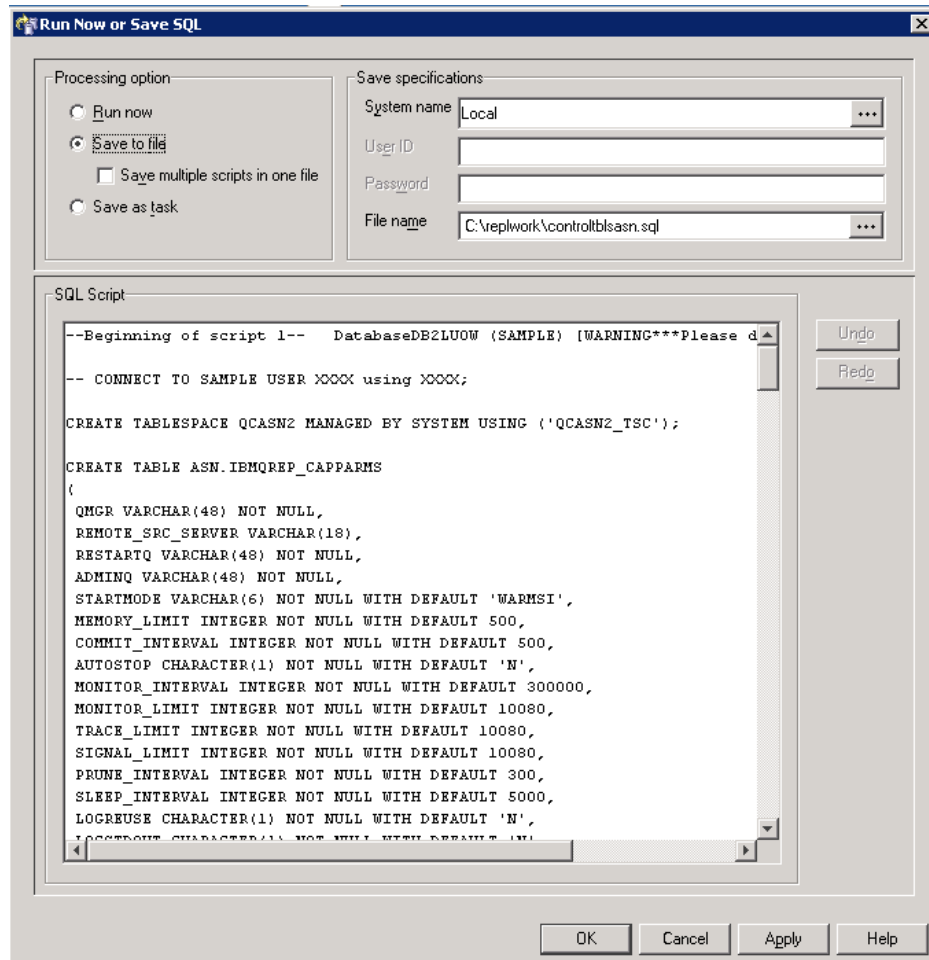


Figure 11-7 Run Now or Save SQL panel for Q Capture Control Tables

## 7. Creating the Q Apply Control tables.

This can be done from the target or source server. If done from the source, make sure that the target server is cataloged in the source.

- a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box.
- b. Click **2. Create Q Apply Control Tables** from the Replication Center Launchpad. This opens the Create Q Apply Control Tables Wizard. From the first panel in the wizard, choose **Typical** or **Custom**, then click **Next**. See Figure 11-8.

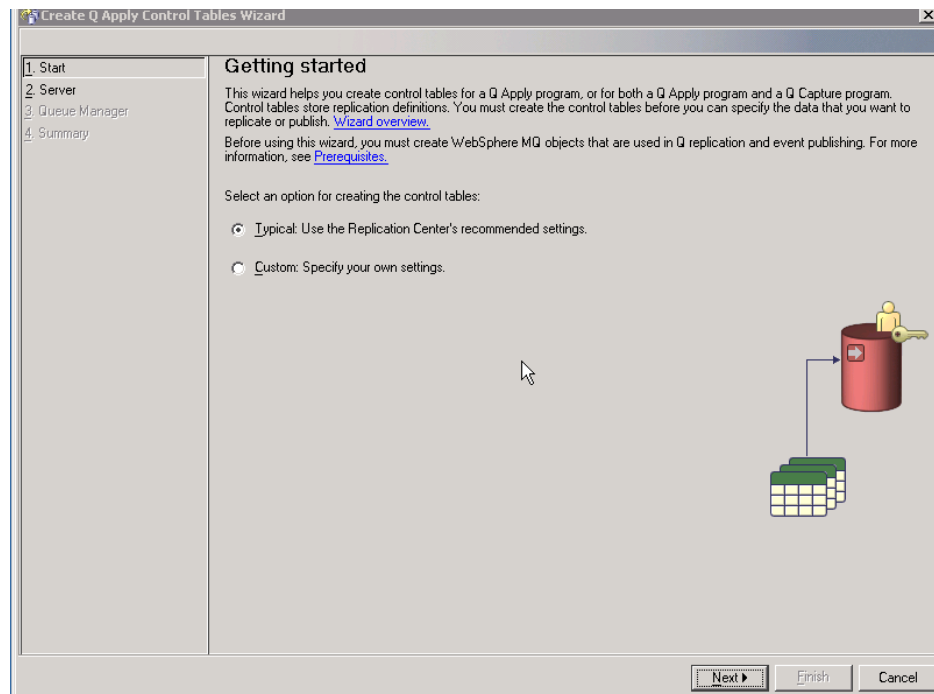


Figure 11-8 Create Q Apply Control Tables Wizard - Getting started

- c. Select the Q Apply server from the list (target server). This should populate the User ID, Password, and Q Apply schema. The default schema is ASN, but can be changed if necessary. Click **Next** to proceed. See Figure 11-9.

**Create Q Apply Control Tables Wizard**

1. Start  
2. **Server**  
3. Queue Manager  
4. Summary

**Specify a Q Apply server, and a Q Apply schema**

Specify the Q Apply server. The Q Apply server is the DB2 database (Linux, UNIX, Windows) or DB2 subsystem (z/OS) where the Q Apply program will run. The control tables will be created on this server. Next, specify a schema to identify the Q Apply program and its unique set of control tables.

Q Apply server: TGTDB

User ID: db2admin

Password: xxxxxxxx

Q Apply schema: ASN

DB2 subsystem:

Database:

Back Next Finish Cancel

Figure 11-9 Create Q Apply Control Tables Wizard panel 2

- d. Insert the Queue manager name for the target server. In our example, the Queue manager is QMGR2. Click **Next** to proceed to the summary panel. See Figure 11-10.

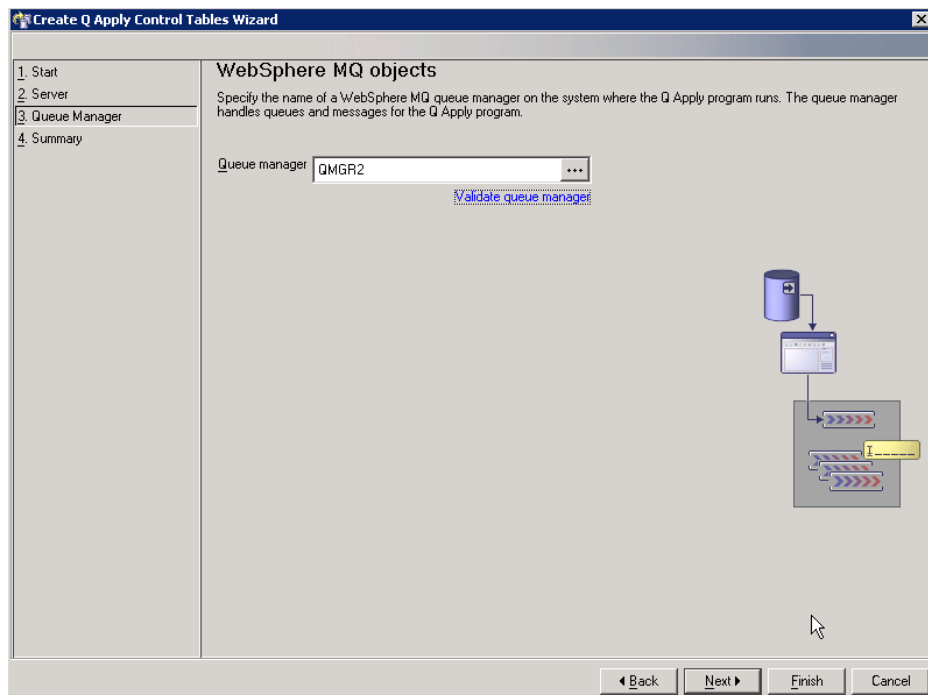


Figure 11-10 Create Q Apply Control Tables Wizard panel 3

e. Review the Summary panel and click **Finish**. See Figure 11-11.

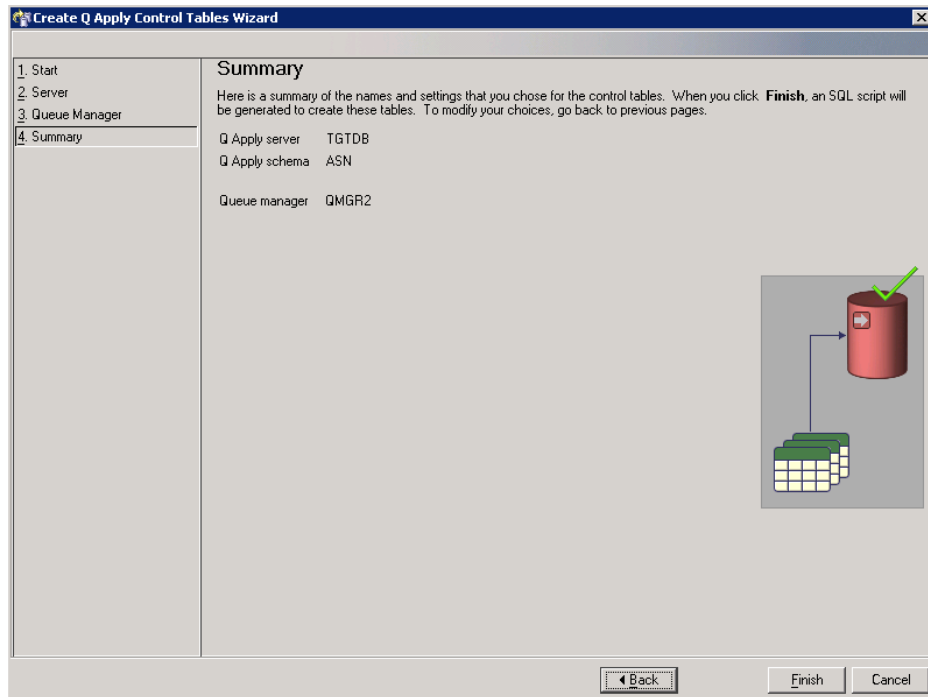


Figure 11-11 Create Q Apply Control Tables Wizard panel 4



- f. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 11-12. After saving the SQL, select **Run now** and click **OK**.

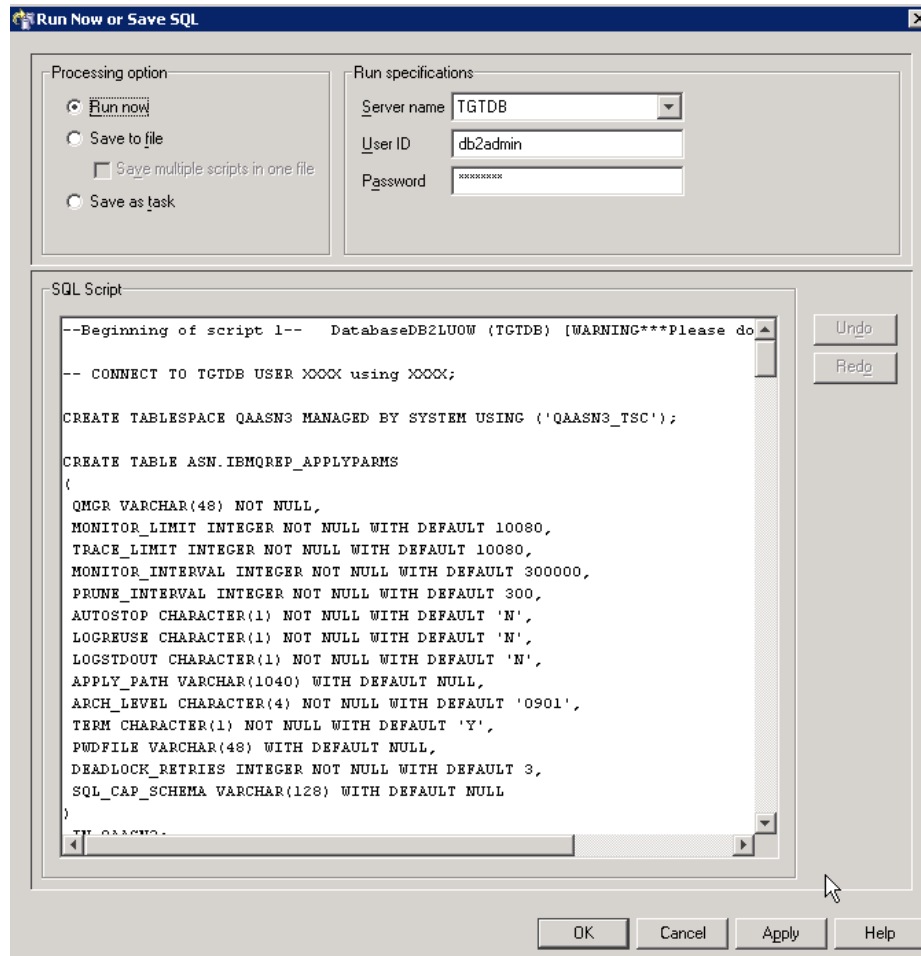


Figure 11-12 Run Now or Save SQL panel for Q Apply Control Tables

8. Create a Q subscription:
  - a. From the Replication Center, open the **Replication Center** drop-down and click **Launchpad**. At the top of the Replication Center Launchpad window, select **Q replication** from the drop-down box.

- b. Click **3. Create a Q Subscription** from the Replication Center Launchpad. This opens the Create a Q Subscription Wizard. From the first panel in the wizard, click **Next**. See Figure 11-13.

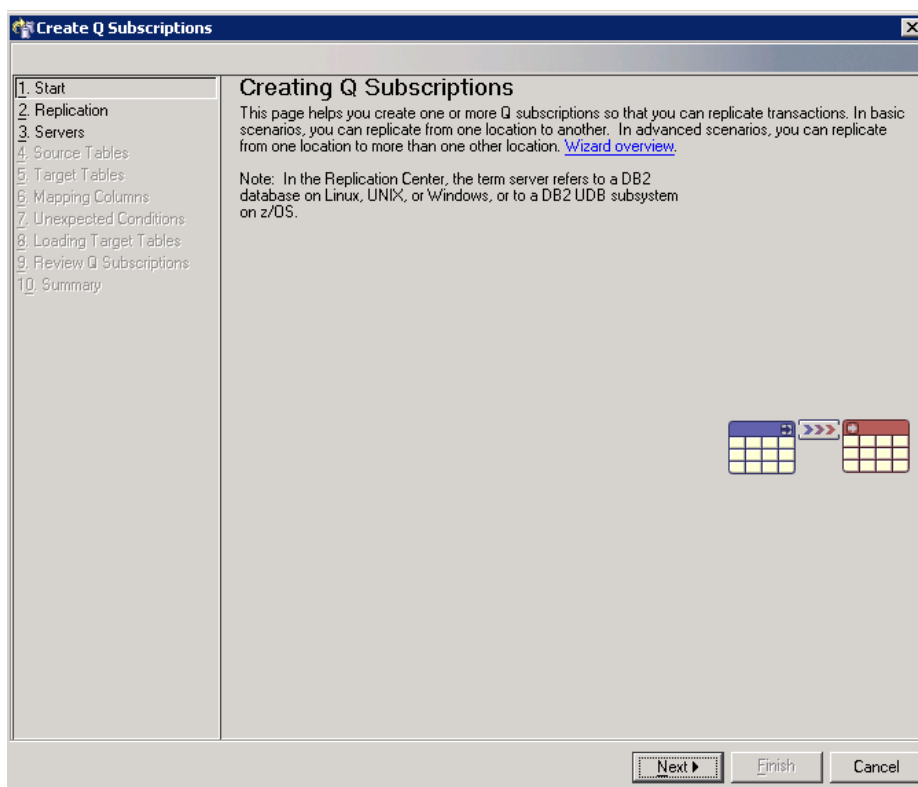


Figure 11-13 Create Q Subscription Wizard Start

- c. Choose the type of Q replication you require. Our example is for unidirectional replication. See Figure 11-14. If you choose anything other than unidirectional, additional MQ setup steps may be required. See DB2V9 Information Center for more information at:

[http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.nav.doc/navcontainers/iiypnavc\\_config\\_q\\_replication.html](http://publib.boulder.ibm.com/infocenter/db2luw/v9/topic/com.ibm.websphere.ii.db2udb.nav.doc/navcontainers/iiypnavc_config_q_replication.html)

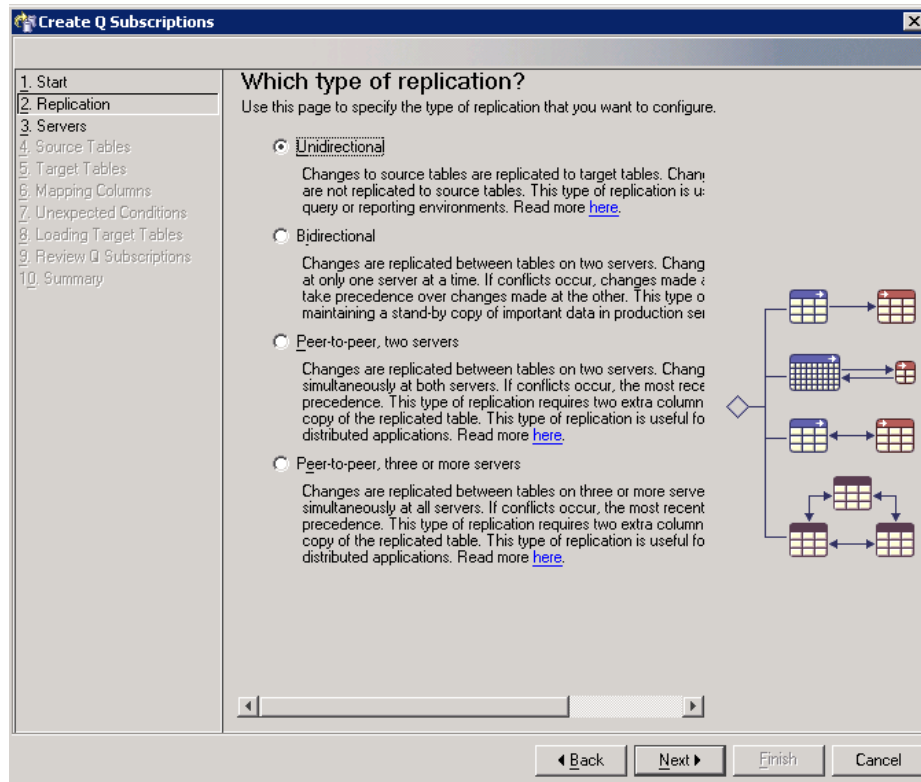


Figure 11-14 Create Q Subscription Wizard Replication

- d. Choose the Source and Target servers and schema if not already populated. In the Queues section, click the button next to the box to open the Select Replication Queue Map window. See Figure 11-15.

The screenshot shows the 'Create Q Subscriptions' wizard window. On the left is a vertical list of steps: 1. Start, 2. Replication, 3. Servers (highlighted), 4. Source Tables, 5. Target Tables, 6. Mapping Columns, 7. Unexpected Conditions, 8. Loading Target Tables, 9. Review Q Subscriptions, and 10. Summary. The main area is titled 'Which source and target servers?' and contains three sections: 'Source', 'Target', and 'Queues'. The 'Source' section has a text box for 'Server' with 'SAMPLE' and a dropdown for 'Schema' with 'ASN'. The 'Target' section has a text box for 'Server' with 'TGTDDB' and a dropdown for 'Schema' with 'ASN'. The 'Queues' section has a text box for 'Replication queue map' and a button with three dots. A yellow tooltip points to the button, stating 'Opens the Select Replication Queue Map window.' At the bottom are buttons for 'Back', 'Next', 'Finish', and 'Cancel'.

**Create Q Subscriptions**

**Which source and target servers?**

Use this page to specify the source and target servers. You must also specify the Q Capture and Q Apply programs to use for capturing data at the source and applying it at the target and a replication queue map that the Q Capture and Q Apply program can use to communicate.

**Source**

This server must contain the tables that you want to replicate.

Server: SAMPLE

Schema: ASN

**Target**

The target tables to which you want to replicate might or might not exist on this server. If they do not, they will be created for you.

Server: TGTDDB

Schema: ASN

**Queues**

Select a replication queue map that the Q Capture and Q Apply program can use to communicate.

Replication queue map: [ ]

Opens the Select Replication Queue Map window.

Back Next Finish Cancel

Figure 11-15 Create Q Subscriptions Wizard Servers before

- e. Click **New** on the Select Replication Queue Map window. See Figure 11-16.

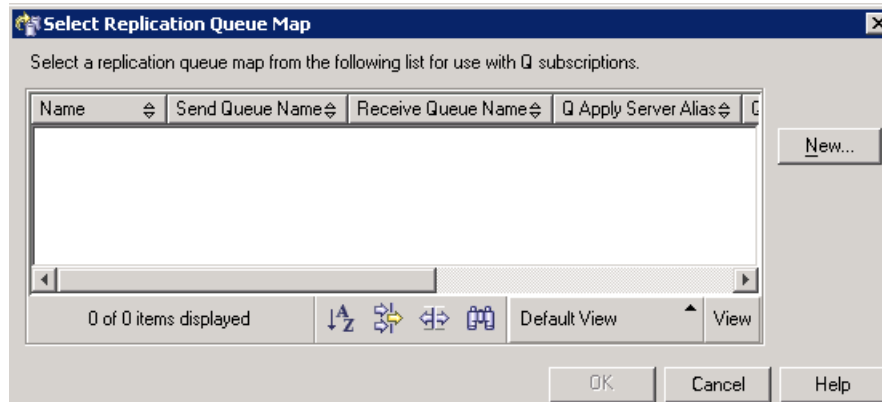


Figure 11-16 Select Replication Queue Map New

- f. In the Create Replication Queue Map window under the General tab, insert the Send queue, Receive queue, and Administration queue. See Figure 11-17.

The screenshot shows the 'Create Replication Queue Map' dialog box with the 'General' tab selected. The dialog box has a title bar with a close button. Below the title bar is a descriptive text: 'Specify the properties for a new replication queue map. A replication queue map specifies which WebSphere MQ message queues to use for a Q subscription. On the Options tab, you can specify how to handle errors, the number of Q Apply agents, and other properties.'

The dialog is divided into two main sections: 'QCapture' on the left and 'QApply' on the right. Each section has a sub-tab labeled 'General' and 'Options'. The 'QCapture' section contains fields for 'Q Capture server' (SAMPLE), 'Q Capture schema' (ASN), 'Send queue' (SAMPLE2TGTDDBQ), and 'Administration queue' (SAMPLE\_ADMINQ). The 'QApply' section contains fields for 'Q Apply server' (TGTDDB), 'Q Apply schema' (ASN), 'Receive queue' (SAMPLE2TGTDDBQ), and 'Administration queue' (SAMPLE\_ADMINQ). Between the two sections, there are two arrows: 'Data Messages' pointing right and 'Administrative Messages' pointing left. At the bottom, there is a field for 'Replication Queue Map Name' (SAMPLE\_ASN\_TO\_TGTDDB\_ASN) and a 'Validate queues' link. The dialog box has 'OK', 'Cancel', and 'Help' buttons at the bottom right.

**Create Replication Queue Map**

Specify the properties for a new replication queue map. A replication queue map specifies which WebSphere MQ message queues to use for a Q subscription. On the Options tab, you can specify how to handle errors, the number of Q Apply agents, and other properties.

**General** Options

**QCapture**

Q Capture server  
SAMPLE

Q Capture schema  
ASN

Send queue  
SAMPLE2TGTDDBQ

Administration queue  
SAMPLE\_ADMINQ

**QApply**

Q Apply server  
TGTDDB

Q Apply schema  
ASN

Receive queue  
SAMPLE2TGTDDBQ

Administration queue  
SAMPLE\_ADMINQ

Data Messages

Administrative Messages

Validate queues

Replication Queue Map Name  
SAMPLE\_ASN\_TO\_TGTDDB\_ASN

OK Cancel Help

Figure 11-17 Create Replication Queue Map General tab

- g. In the Create Replication Queue Map window under the Options tab, you can specify the attributes for the new replication queue map. For our example, we kept the default values. Click **OK** to finish the new replication queue map. See Figure 11-18.

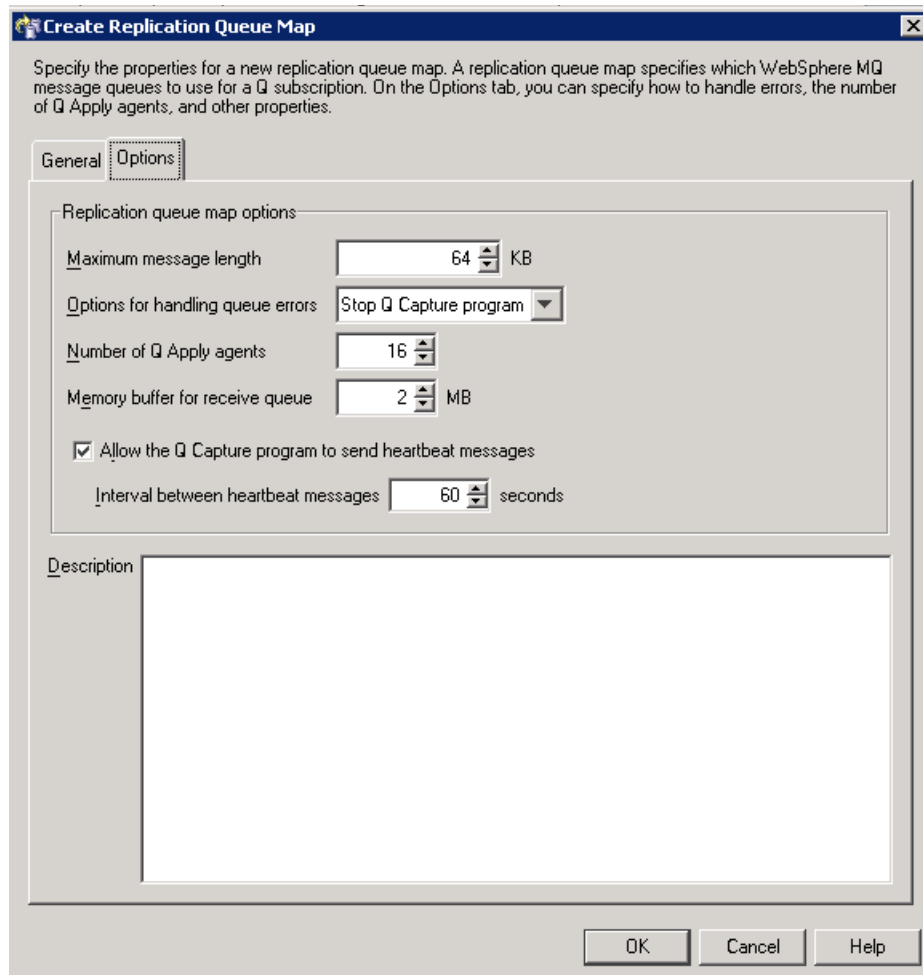
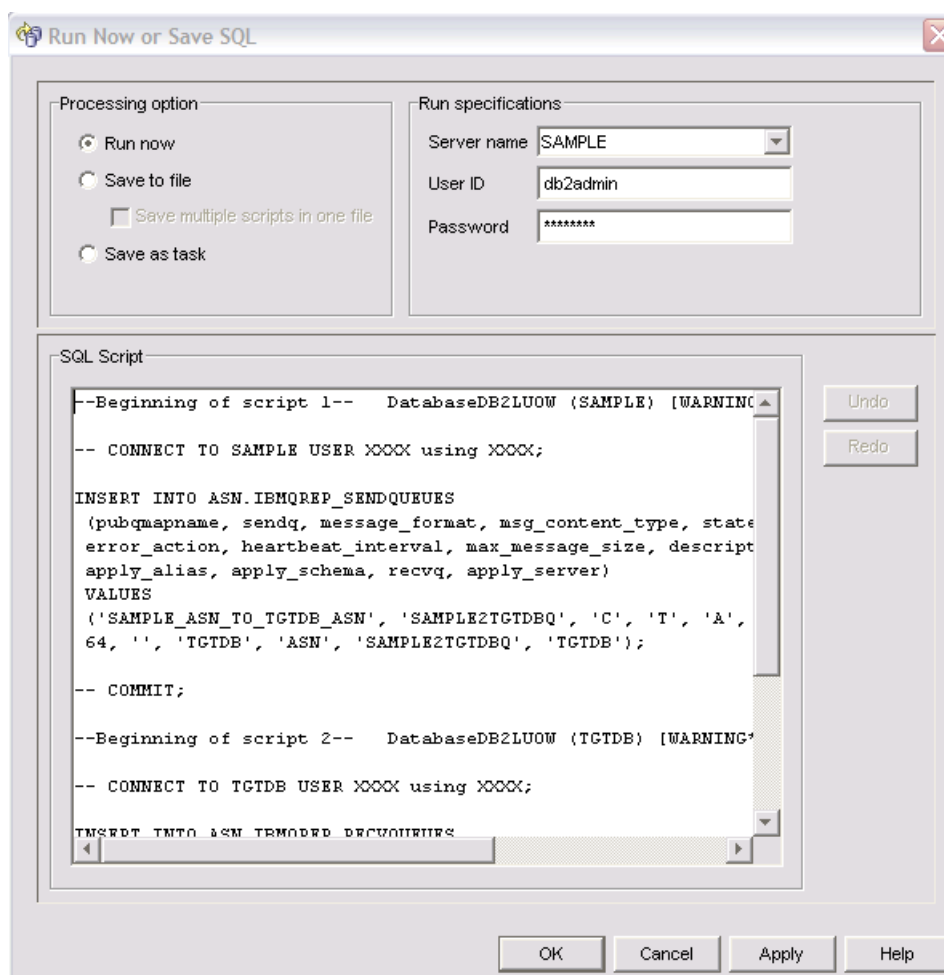


Figure 11-18 Create Replication Queue Map Options tab

- h. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 11-19. After saving the SQL, select **Run now** and click **OK**.



The image shows a 'Run Now or Save SQL' dialog box. It has two main sections: 'Processing option' and 'Run specifications'. The 'Processing option' section has four radio buttons: 'Run now' (selected), 'Save to file', 'Save multiple scripts in one file' (disabled), and 'Save as task'. The 'Run specifications' section has three text fields: 'Server name' (SAMPLE), 'User ID' (db2admin), and 'Password' (masked with asterisks). Below these is a large text area for the 'SQL Script' containing two SQL scripts. The first script is for 'DatabaseDB2LUOW (SAMPLE)' and the second is for 'DatabaseDB2LUOW (TGTDDB)'. To the right of the text area are 'Undo' and 'Redo' buttons. At the bottom are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

Run Now or Save SQL

Processing option

☒ Run now

☐ Save to file

☐ Save multiple scripts in one file

☐ Save as task

Run specifications

Server name: SAMPLE

User ID: db2admin

Password: \*\*\*\*\*

SQL Script

```
--Beginning of script 1-- DatabaseDB2LUOW (SAMPLE) [WARNING]
-- CONNECT TO SAMPLE USER XXXX using XXXX;

INSERT INTO ASN.IBMQREP_SENDQUEUES
(pubqmapname, sendq, message_format, msg_content_type, state,
error_action, heartbeat_interval, max_message_size, descriptor,
apply_alias, apply_schema, recvq, apply_server)
VALUES
('SAMPLE_ASN_TO_TGTDDB_ASN', 'SAMPLE2TGTDDBQ', 'C', 'T', 'A',
64, '', 'TGTDDB', 'ASN', 'SAMPLE2TGTDDBQ', 'TGTDDB');

-- COMMIT;

--Beginning of script 2-- DatabaseDB2LUOW (TGTDDB) [WARNING]
-- CONNECT TO TGTDDB USER XXXX using XXXX;

INSERT INTO ASN.IBMQREP_SENDQUEUES
```

Undo

Redo

OK Cancel Apply Help

Figure 11-19 Run Now or Save SQL Queue Map



- i. From the Select Replication Queue Map, highlight your newly created queue map, then click **OK**. See Figure 11-20.

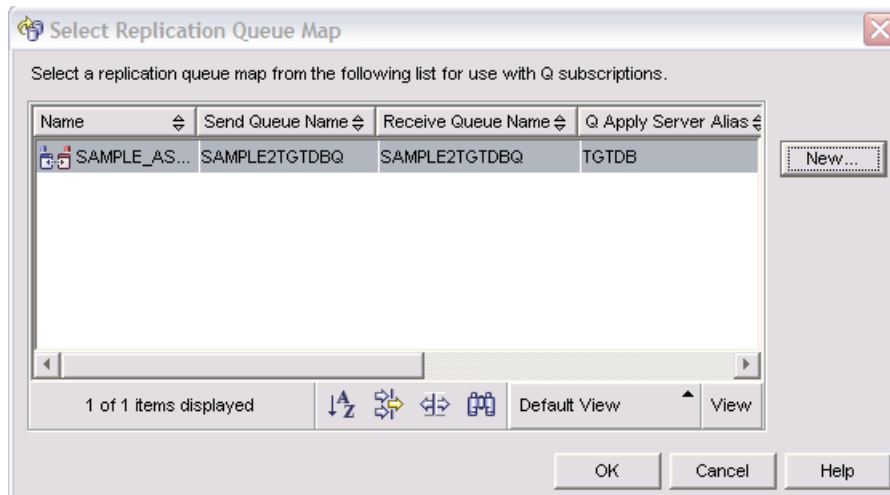


Figure 11-20 Select Replication Queue Map selected

- j. Now you should be back to the Create Q Subscription wizard panel with all of the information filled in for 3. Servers. See Figure 11-21. Click **Next** to continue.

The screenshot shows the 'Create Q Subscriptions' wizard window. On the left is a vertical list of steps from 1 to 10. Step 3, 'Servers', is highlighted. The main area is titled 'Which source and target servers?' and contains instructions: 'Use this page to specify the source and target servers. You must also specify the Q Capture and Q Apply programs to use for capturing data at the source and applying it at the target and a replication queue map that the Q Capture and Q Apply program can use to communicate.'

There are three sections in the main area:

- Source:** 'This server must contain the tables that you want to replicate.' It has a 'Server' field with 'SAMPLE' and a 'Schema' dropdown menu with 'ASN' selected.
- Target:** 'The target tables to which you want to replicate might or might not exist on this server. If they do not, they will be created for you.' It has a 'Server' field with 'TGTDDB' and a 'Schema' dropdown menu with 'ASN' selected.
- Queues:** 'Select a replication queue map that the Q Capture program and Q Apply program can use to communicate.' It has a 'Replication queue map' field with 'SAMPLE\_ASN\_TO\_TGTDDB\_ASN'.

At the bottom right are four buttons: 'Back', 'Next', 'Finish', and 'Cancel'.

Figure 11-21 Create Q Subscriptions panel 3 complete

- k. Select the tables from the source you wish to replicate. In our example, we chose one table to replicate. See Figure 11-22. Click **Next** to continue.

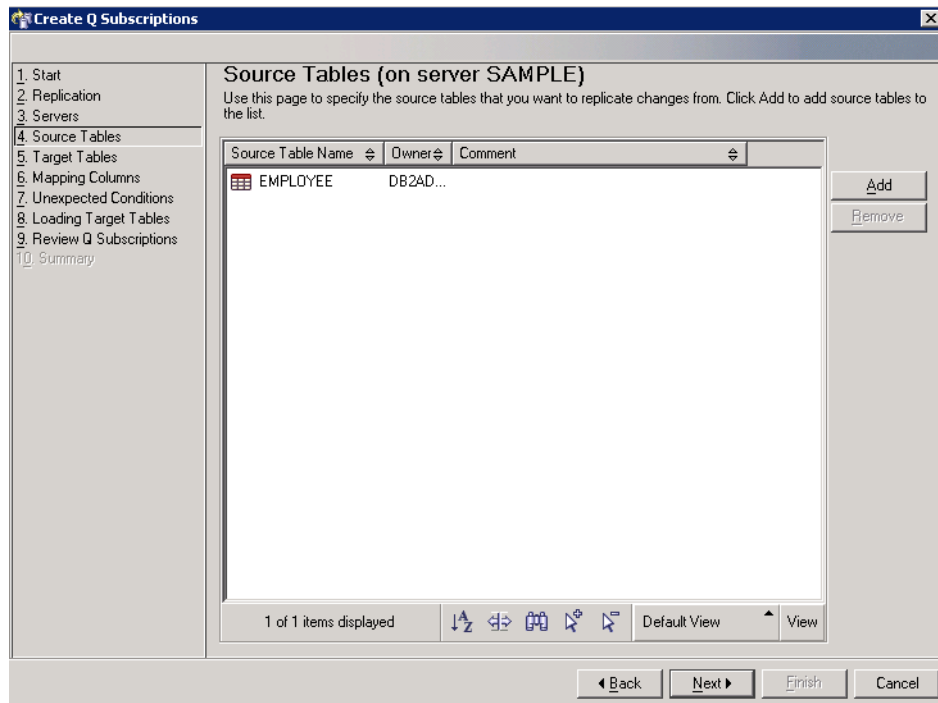


Figure 11-22 Create Q Subscriptions source tables

- I. For step 5. Target, in the Create Q Subscriptions wizard, select the target type you want. For our example, we chose to create a new table on the target. See Figure 11-23. Click **Next** to continue.

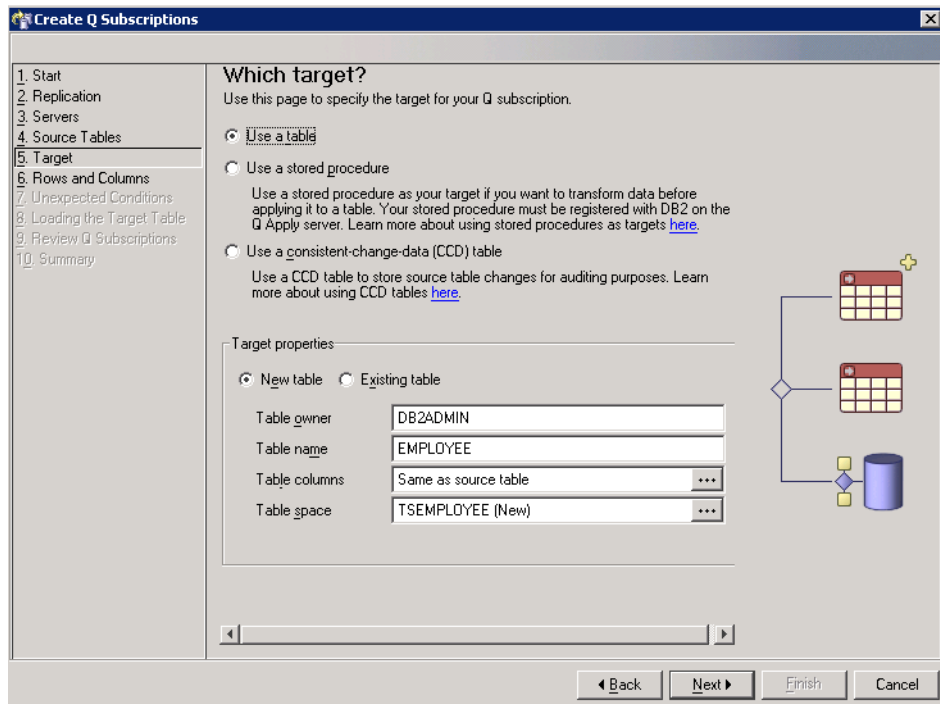


Figure 11-23 Create Q Subscriptions target tables

- m. For step 6. Rows and Columns, in the Create Q Subscriptions wizard, you have the choice of granularity with your columns and rows replicated. See Figure 11-24. Click **Next** to continue.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target  
6. Rows and Columns  
7. Unexpected Conditions  
8. Loading the Target Table  
9. Review Q Subscriptions  
10. Summary

### Which columns and rows should be replicated?

Use this page to specify the subset of data that you want to replicate. You do this by specifying columns and rows of the source table. You must also map selected source columns to target columns if you are replicating to a target table, or map selected source columns to target parameters if you are replicating to a stored procedure. You must also specify a set of key columns that the Q Apply program can use to identify target rows. The Q Apply program uses the key columns to make sure that transactions that change the same rows are applied to the target tables or stored procedures in order.

**Source changes**

Which source changes should the Q Capture program replicate?

Columns: All columns ...

Rows: All rows ...

**Column mapping**

How should the source columns be mapped to the target columns?

Column mapping: Mapped by name ...

**Target index or key**

How should the Q Apply program ensure that transactions are applied in the right order?

Index or primary key: Primary key ...

◀ Back   Next ▶   Finish   Cancel

Figure 11-24 Create Q Subscriptions replicated columns and rows

- n. For step 7. Unexpected Conditions, in the Create Q Subscriptions wizard, you can specify how you want the Q Apply program to handle unexpected conditions. See Figure 11-25. Click **Next** to continue.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target  
6. Rows and Columns  
7. Unexpected Conditions  
8. Loading the Target Table  
9. Review Q Subscriptions  
10. Summary

### How should the Q Apply program respond to unexpected conditions?

Use this page to specify how the Q Apply program should respond to unexpected conditions.

The Q Apply program updates target tables with changes replicated from your source tables. If other applications are also making changes to your target tables, the Q Apply program might discover rows in the target tables that are different than it expects. For example, the Q Apply program might try to update a row that another application already deleted. Read more examples [here](#).

Unexpected conditions in the target data

How should the Q Apply program respond to unexpected conditions in target data?

- ☐ Force the source change into the target table
- ☒ Do not force the source change into the target table
  - ☒ Ignore the condition and continue
  - ☐ Stop the corresponding Q subscription
  - ☐ Stop the corresponding receive queue
  - ☐ Shutdown

The Q Apply program might encounter other types of unexpected conditions while applying data to target tables, such as SQL errors.

Other unexpected conditions

How should the Q Apply program respond to these other unexpected conditions?

- ☐ Stop the corresponding Q subscription
- ☒ Stop the corresponding receive queue
- ☐ Shutdown

◀ Back   Next ▶   Finish   Cancel

Figure 11-25 Create Q Subscriptions Unexpected Conditions

- o. For step 8. Loading the Target Table, specify how you want your target tables loaded. See Figure 11-26. Click **Next** to continue.

**Create Q Subscriptions**

1. Start  
2. Replication  
3. Servers  
4. Source Tables  
5. Target  
6. Rows and Columns  
7. Unexpected Conditions  
8. Loading the Target Table  
9. Review Q Subscriptions  
10. Summary

### How should the target table be loaded?

Use this page to specify if and how your target table will be loaded. When the Q subscription starts, the Q Apply program can initiate a load of the target table with the utility or utilities that you specify.

Loading the target table

How will the target table be loaded when this Q subscription starts?

☒ Automatic: The Q Apply program performs the load

How should the Q Apply program choose which load method to use?

☐ Best available: Let the Q Apply program choose a load method

☐ Always use LOAD FROM CURSOR

☒ Always use export and load

☐ Always use export and import

☐ Manual: You perform the load manually and inform the Q Apply program when the load is complete

☐ None: The target table will not be loaded

Nickname

Which nickname should Q Apply use to load rows from the source table?

☒ Create a new nickname

Nickname owner:

Nickname name:

Server definition:

☐ Use an existing nickname

Back Next Finish Cancel

Figure 11-26 Create Q Subscription Loading the Target Table

- p. For step 9. Review Q Subscriptions, in the Create Q Subscriptions wizard, observe the settings. See Figure 11-27. Click **Next** to continue.

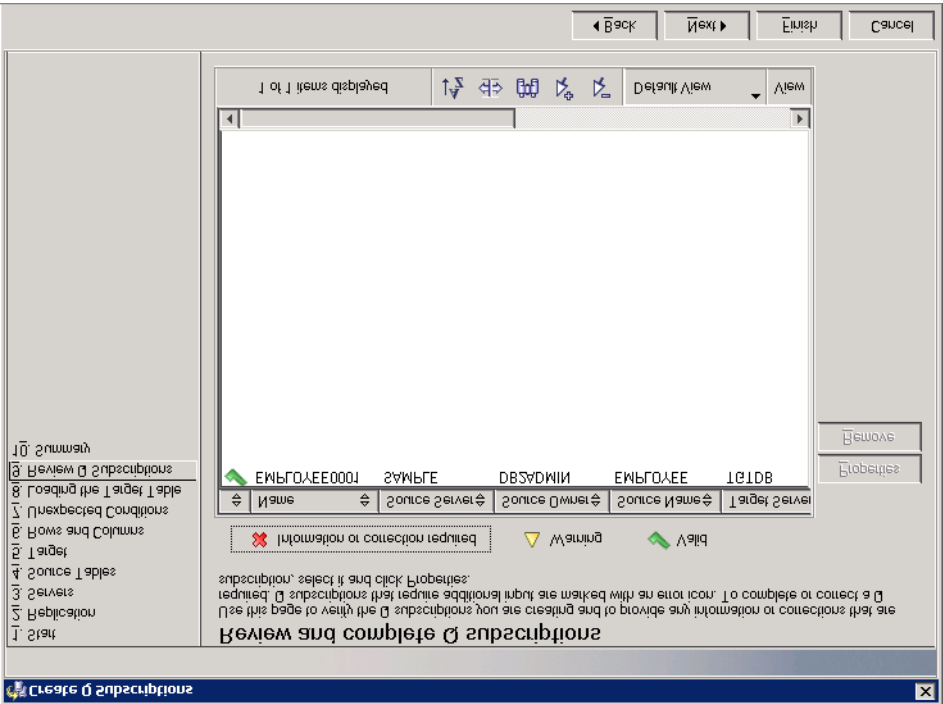


Figure 11-27 Create Q Subscriptions, Review Q Subscriptions



- q. Step 10. Summary, in the Create Q Subscriptions wizard, is the final window before creating the SQL. See Figure 11-28. Click **Finish** to create the SQL.

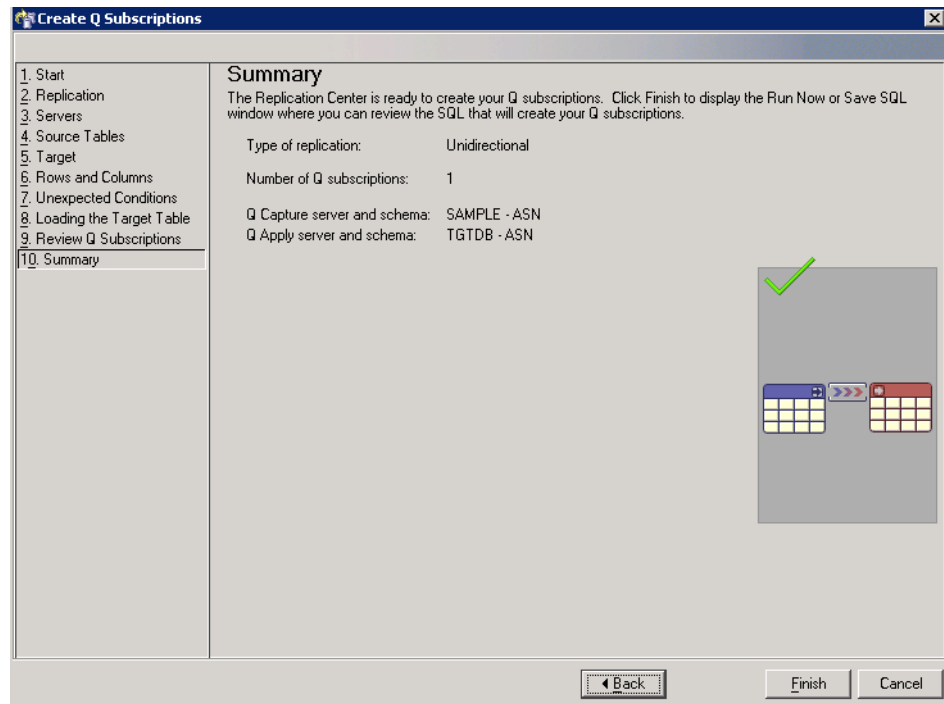
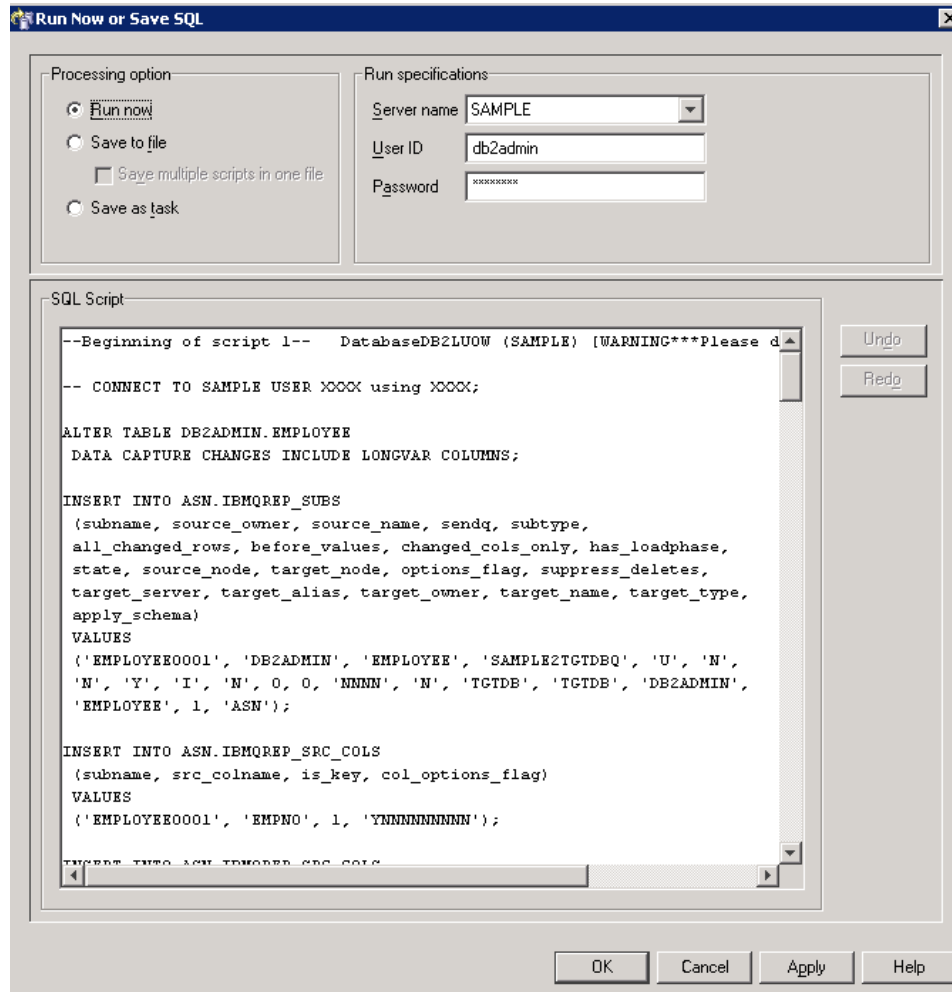


Figure 11-28 Create Q Subscription Summary

- r. In the Run Now or Save SQL panel, we recommend that you save the script for future use, such as a rebuild. See Figure 11-29. After saving the SQL, select **Run now** and click **OK**.



The screenshot shows the 'Run Now or Save SQL' dialog box. It has a title bar with a small icon and the text 'Run Now or Save SQL'. The dialog is divided into several sections. The 'Processing option' section on the left has four radio buttons: 'Run now' (selected), 'Save to file', 'Save multiple scripts in one file' (disabled), and 'Save as task'. The 'Run specifications' section on the right has three text boxes: 'Server name' (containing 'SAMPLE'), 'User ID' (containing 'db2admin'), and 'Password' (containing 'XXXXXXXX'). Below these is the 'SQL Script' section, which contains a text area with a script. The script starts with a comment '--Beginning of script 1-- Database DB2LUOW (SAMPLE) [WARNING\*\*\*Please d' and continues with SQL commands: '-- CONNECT TO SAMPLE USER XXXX using XXXX;', 'ALTER TABLE DB2ADMIN.EMPLOYEE DATA CAPTURE CHANGES INCLUDE LONGVAR COLUMNS;', 'INSERT INTO ASN.IBMQREP\_SUBS (subname, source\_owner, source\_name, sendq, subtype, all\_changed\_rows, before\_values, changed\_cols\_only, has\_loadphase, state, source\_node, target\_node, options\_flag, suppress\_deletes, target\_server, target\_alias, target\_owner, target\_name, target\_type, apply\_schema) VALUES ('EMPLOYEE0001', 'DB2ADMIN', 'EMPLOYEE', 'SAMPLE2TGTDBQ', 'U', 'N', 'N', 'Y', 'I', 'N', 0, 0, 'NNNN', 'N', 'TGTDB', 'TGTDB', 'DB2ADMIN', 'EMPLOYEE', 1, 'ASN');', 'INSERT INTO ASN.IBMQREP\_SRC\_COLS (subname, src\_colname, is\_key, col\_options\_flag) VALUES ('EMPLOYEE0001', 'EMPNO', 1, 'YNNNNNNNNN');', and 'INSERT INTO ASN.IBMQREP\_SRC\_COLS'. To the right of the text area are 'Undo' and 'Redo' buttons. At the bottom of the dialog are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

Figure 11-29 Run Now or Save SQL Q Subscriptions

9. Configure target to source server connectivity.

If you plan to have the Q Apply program automatically load targets with source data by using the EXPORT utility, as we did in our example, the Q Apply program must be able to connect to the Q Capture server. This connection requires a password file that is created with the **asnpwd** command.

- a. Create a directory on the Q Apply Control server where you plan to run the Q Apply start command. In our example, we used `c:\replwork\`.
- b. Run the command to create the password file in the directory that you have created. See Example 11-5.

**asnpwd init**

*Example 11-5 asnpwd init*

---

```
C:\replwork> asnpwd init
2006-10-25-16.31.47.671000 ASN1981I  "Asnpwd" : "" : "Initial". The
program completed successfully using password file "asnpwd.aut".
```

---

- c. Run the command to update the `asnpwd.aut` file with the connection information. See Example 11-6.

**asnpwd add alias SAMPLE id db2admin password adminpw**

*Example 11-6 asnpwd add*

---

```
C:\replwork> asnpwd add alias SAMPLE id db2admin password adminpw
2006-10-25-16.37.10.535000 ASN1981I  "Asnpwd" : "" : "Initial". The
program completed successfully using password file "asnpwd.aut".
```

---

The unidirectional configuration has been completed. We can start the Q replication process. When start Q replication, start Q Capture first then Q Apply. To stop the replication process, stop Q Apply then Q Capture.

## Start Q Capture

This can be done either from the command line or from the Replication Center.

### ***Command line***

To start Q Capture from the command line, change to the desired directory and issue the start command. See Example 11-7.

**asnqcap capture\_server=SAMPLE**

*Example 11-7 asnqcap*

---

```
C:\replwork> asnqcap capture_server=SAMPLE
2006-10-25-16.49.51.309000 ASN7000I  "Q Capture" : "ASN" :
"WorkerThread" : "1" subscriptions are active. "0" subscriptions are
inactive. "0" subscriptions that were new and were successfully
activated. "0" subscriptions that were new could not be activated and
are now inactive.
2006-10-25-16.49.51.329000 ASN0572I  "Q Capture" : "ASN" :
"WorkerThread" : The program initialized successfully.
```

---

## Replication Center

To start Q Capture, complete the following steps:

1. In the Replication Center, expand **Q Replication** → **Operations** → **Q Capture Servers**. In the right pane, right-click the server on which you are starting Q Capture. See Figure 11-30.

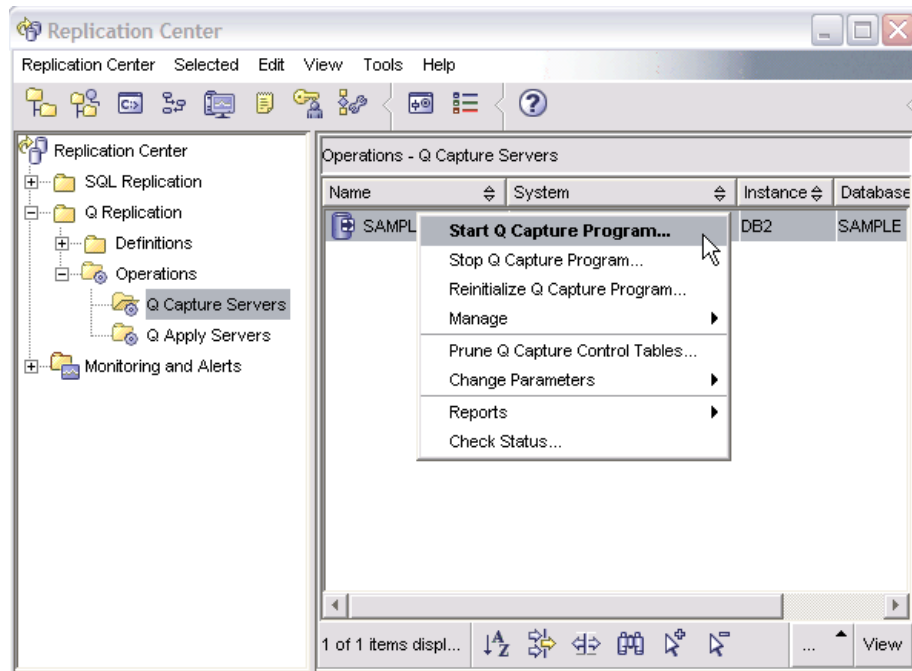


Figure 11-30 Replication Center Start Q Capture

2. From the Run Now or Save Command window, insert the required information in to the User ID, Password and start Directory fields. Click **OK** to issue the command. See Figure 11-31.

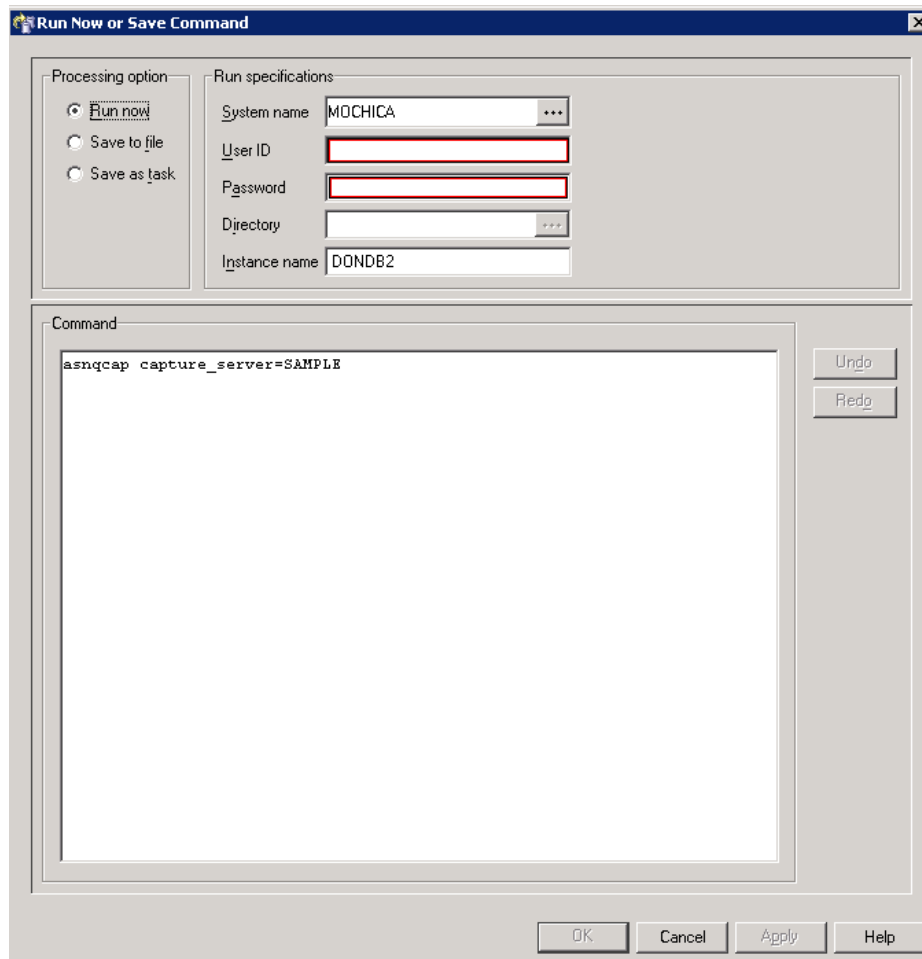


Figure 11-31 Run Now or Save Command Q Capture Start

## Start Q Apply

This can be done from either the command line or from the Replication Center.

## Command line

To start Q Apply from the command line, change to the desired directory and issue the start command. See Example 11-8.

**asnqapp apply\_server=TGTDDB**

*Example 11-8 asnqapp*

```
C:\replwork> asnqapp apply_server=TGTDDB
2006-10-25-15.58.20.785000 ASN7526I  "Q Apply" : "ASN" : "BR00000" :
The Q Apply program has started processing the receive queue
"SAMPLE2TGTDDBQ" for replication queue map "SAMPLE_ASN_TO_TGTDDB_ASN".
```

## Replication Center

To start Q Apply, use the following steps:

1. In the Replication Center, expand **Q Replication** → **Operations** → **Q Apply Servers**. In the right pane, right-click the server on which you are starting Q Apply. See Figure 11-32.

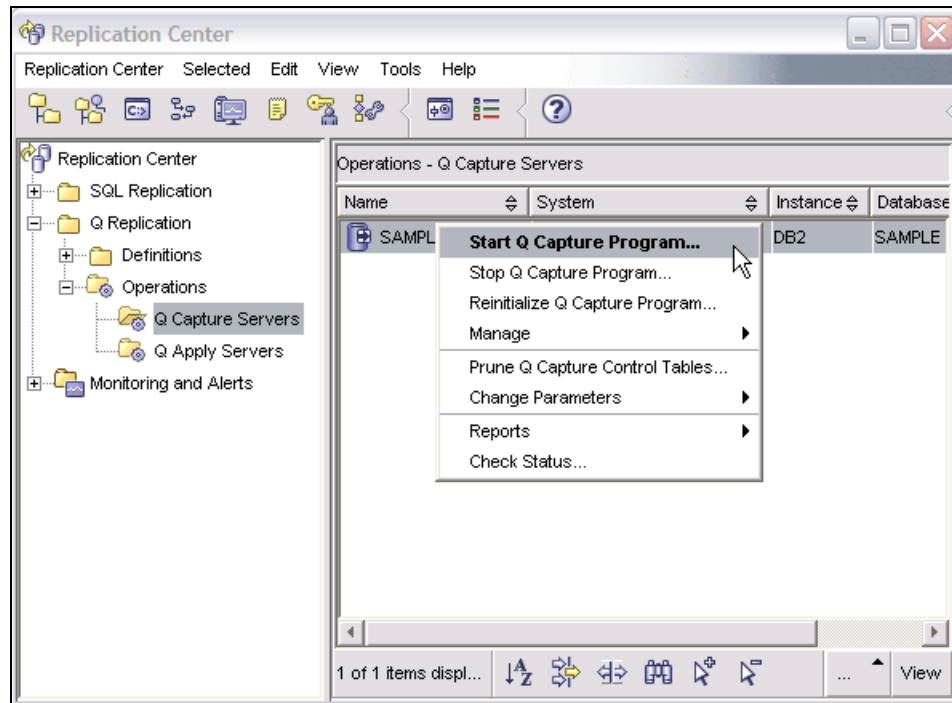
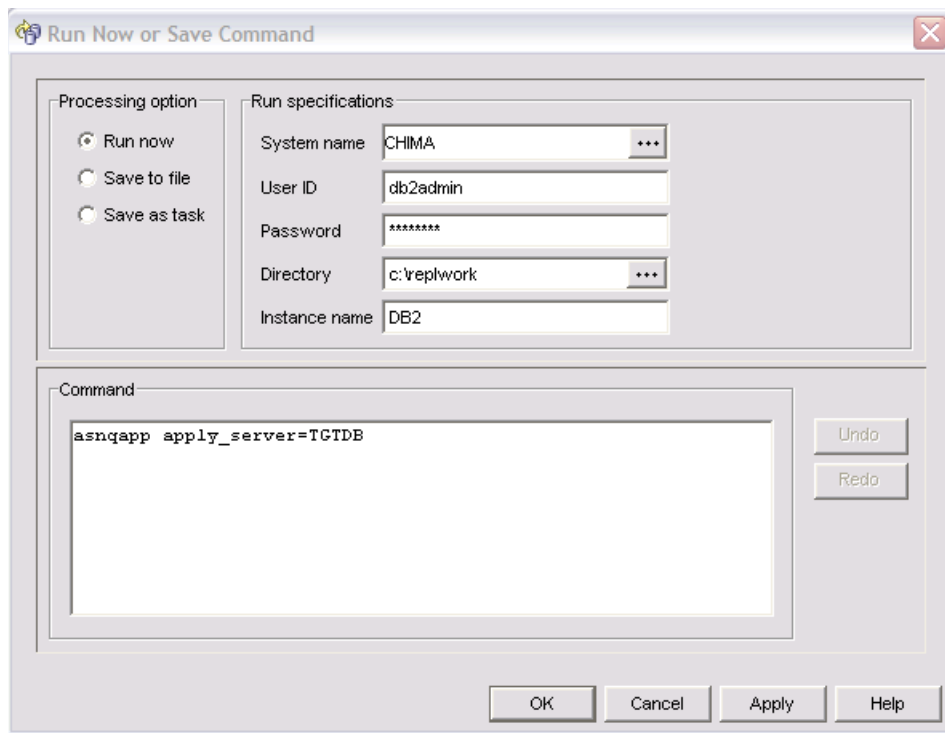


Figure 11-32 Replication Center Start Q Apply

2. From the Run Now or Save Command window, insert the required information into the User ID, Password and start Directory fields. Click **OK** to issue the command. See Figure 11-33.



The dialog box titled "Run Now or Save Command" contains two main sections. The "Processing option" section on the left has three radio buttons: "Run now" (selected), "Save to file", and "Save as task". The "Run specifications" section on the right contains five text input fields: "System name" (CHIMA), "User ID" (db2admin), "Password" (masked with asterisks), "Directory" (c:\replwork), and "Instance name" (DB2). Below these is a "Command" section with a large text area containing the command "asncapp apply\_server=TGTDB". To the right of the command area are "Undo" and "Redo" buttons. At the bottom of the dialog are "OK", "Cancel", "Apply", and "Help" buttons.

Processing option	Run specifications
<input checked="" type="radio"/> Run now	System name: CHIMA
<input type="radio"/> Save to file	User ID: db2admin
<input type="radio"/> Save as task	Password: *****
	Directory: c:\replwork
	Instance name: DB2

Command: asncapp apply\_server=TGTDB

Buttons: OK, Cancel, Apply, Help

Figure 11-33 Run Now or Save Command Start Q Apply





## Part 2

# DB2 scalability options

In this part of the book, we introduce the DB2 Data Partition Feature (DPF), including an overview, installation steps, and an explanation of how to scale the database. This gives you a quick start in understanding DB2 DPF. We also discuss the DB2 table partitioning feature introduced in DB2 9.





## DB2 Database Partitioning Feature

DB2 for Linux, UNIX, and Windows is highly scalable and capable of supporting hardware platforms from uniprocessor mobile computers, single and clustered Symmetric Multiprocessing (SMP) servers, to parallel systems with hundreds of nodes and many processors per node. This extremely versatile capability provides both extensive and granular growth. In this chapter we introduce the Database Partitioning Feature (DPF) which enables DB2 to scale out to support very large databases and increased parallelism for various tasks.

We discuss the following topics:

- ▶ Overview of DPF
- ▶ Installation
- ▶ Scaling the database
- ▶ High availability (HA) on DPF

## 12.1 Overview of DPF

A hardware trend that we observe is to build enterprise systems by assembling together a group of cheaper, smaller servers. Customers want their databases to take advantage of this trend as their data keeps growing and it is very difficult for them to estimate how big their database will be a few years from now. Customers would like the database architecture to be flexible enough to take advantage of the processing power of another server node being added to an existing cluster.

Database Partitioning Feature (DPF) available on DB2 for LUW Enterprise Server Edition (ESE) extends the capability of this relational database manager into the parallel, multi-node environment. With DPF your database is scalable, as you can add new servers and spread your database across them. This means more CPUs, more memory, and more disks from each of the additional servers for your database. DB2 ESE with DPF is ideal for managing data warehousing, data mining, and online analytical processing (OLAP) workloads. It can also work well with online transaction processing (OLTP) workloads. Figure 12-1 illustrates DB2 partitioned on networked machines. An application can connect to any of the database partitions.

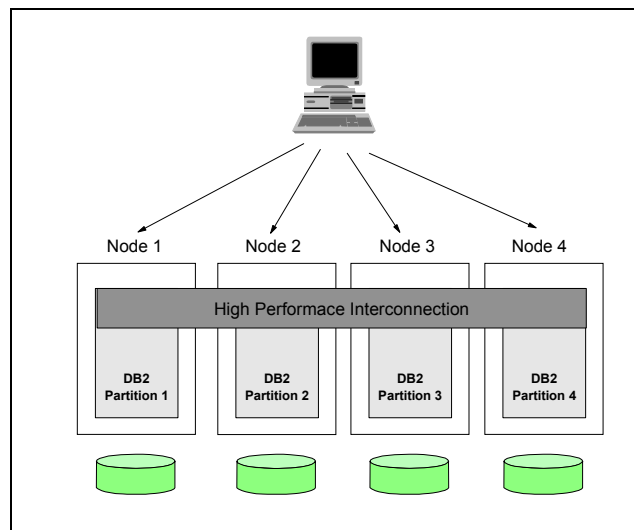
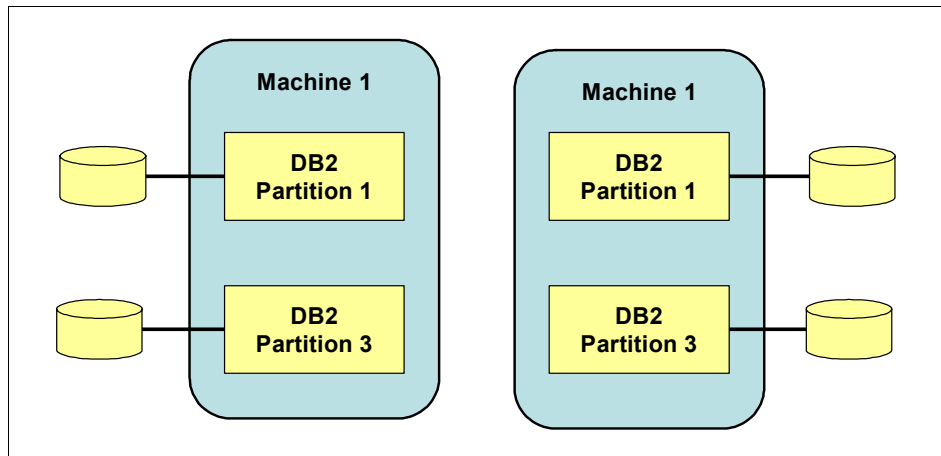


Figure 12-1 DB2 database with database partitions on different servers

DPF allows a DB2 database to have multiple database partitions. When a database is partitioned, the database is split into different independent parts, each consisting of its own data in table space containers, configuration files, indexes, and transaction logs. Multiple partitions can be assigned to a single

physical server and these partitions share the resources of that server. A database partition is sometimes called a node, in a vaguely similar sense to the way a server within a cluster is called a node, although non-replicated partitioned databases require all nodes to be active at once. Figure 12-2 illustrates DB2 partitioned on SMP machines.



*Figure 12-2 Multiple DB2 partitions on one SMP server*

DB2 implements DPF using a shared nothing architecture. The key advantage of the shared nothing architecture is that no special hardware is required and each server only needs to do part of the work, as a result of which there are performance gains from not sharing resources across the network. The shared nothing architecture eliminates the overhead of distributed lock management. The database size is scalable with the number of disks and number of processors. DPF has a very scalable architecture and has minimal communication between the nodes. The DPF shared nothing architecture enables DB2 to provide the best scalability and price, performance or both. Figure 12-3 illustrates this architecture.

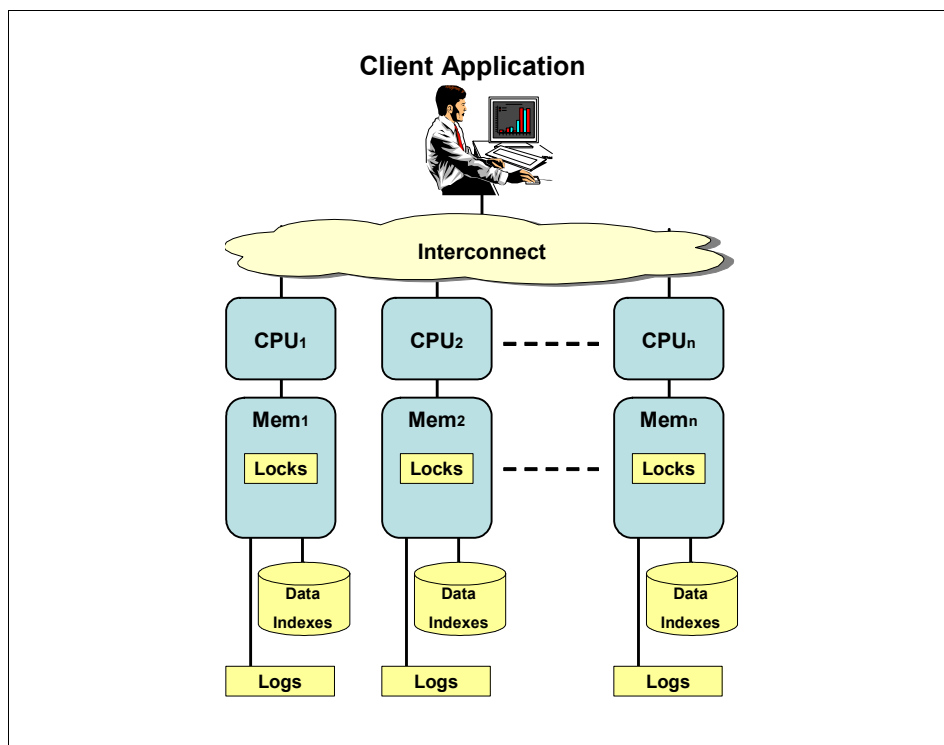


Figure 12-3 Each Partition has its own data, buffer pool and logs

A *single-partition database* is by definition, a database that has only one database partition. All data in the database is stored in that partition. A *multi-partitioned database* is a database with two or more database partitions. Because data is divided across database partitions, you can use the power of multiple processors on multiple physical nodes to satisfy requests for information. DB2 supports a partitioned storage model, in which the <sup>1</sup> (partitioning key) is used to partition table data across a set of database partitions. This means that the data is physically stored across more than one database partition, and yet can be accessed as though it were located in the same place. Applications and users accessing data in a partitioned database need not be aware of the physical location of the data. The data, while physically split, is used and managed as a logical whole. Data retrieval and update requests are decomposed automatically into sub-requests, and executed in parallel among the applicable database partitions. The fact that databases are split across database partitions is transparent to users issuing SQL statements. It is possible, if DB2 determines that a statement or command affects only one, or

<sup>1</sup> In DB2 9.1, the term partitioning key is changed to distribution key.

a subset of all of the database partitions, that DB2 will direct the work to only that partition or that subset of partitions.

User interaction occurs through one database partition, known as the coordinator node for that user. An application can connect to any partition. The partition to which the application is connected acts as the coordinator for that application, and the coordinator is responsible for passing requests and data between the application and the partitions in the database.

Users have the flexibility to spread the workload across a partitioned database for large tables, while allowing smaller tables to be stored on one or more database partitions. Index data is also partitioned with its corresponding tables. Each database partition has local indexes on the data it stores. This results in increased performance for local data access in each partition.

DB2 DPF has the capability of recognizing that data being accessed for a join or a subquery is located on the same partition in the same database partition group. This is known as *table collocation*. Rows in collocated tables with the same distribution key values are located on the same partition. DB2 can choose to perform join or subquery processing at the partition in which the data is stored. This has significant performance advantages. To take advantage of this feature small or medium size tables can be replicated across multiple partitions so that query processing related to them can be done locally.

### 12.1.1 DB2 partitioned instance

Every DB2 partitioned instance is owned by an instance owner and is distinct from other instances. A DB2 instance is created on any one of the machines in the configuration, which becomes the “primary machine”. This primary server is known as the DB2 instance-owning server as its disk physically stores the instance home directory. This instance home directory is exported to the other servers in the DPF environment. On the other servers, a DB2 instance is separately created: all using the same characteristics, the same instance name, the same password, and a shared instance home directory. Each instance can manage multiple databases; however, a single database can only belong to one instance. It is possible to have multiple DB2 instances on the same group of parallel servers.

All the servers participating in a DPF instance must be interconnected by a communication facility which could be a network using the TCP/IP protocol. Several TCP/IP ports are reserved on each server for this “interpartition” communication. The Fast Communication Manager (FCM) will use these ports to handle most of the communication between partitions.

## 12.1.2 The nodes configuration file

The *db2nodes.cfg* file is used to define the database partition servers that will participate in a DB2 instance. The *db2nodes.cfg* file is also used to specify the IP address or host name of a high-speed interconnect, should you want to use a high-speed interconnect for database partition server communication.

The *db2nodes.cfg* file must be located under the `SQLLIB` directory for the instance owner on Linux and UNIX. On Windows, it is located under the `SQLLIB\<Instance name>` directory.

You can update the *db2nodes.cfg* file using an editor, but on a Windows platform it is recommended that the *db2nodes.cfg* file be changed through `db2` commands and never be manually edited.

On Windows platforms, do not attempt to create or modify the node configuration *db2nodes.cfg* file manually. You can use the **db2ncrt** command to add a database partition server to an instance. You can use the **db2ndrop** command to drop a database partition server from an instance. You can use the **db2nchg** command to modify the configuration of a database partition server, including moving the database partition server from one computer to another; changing the TCP/IP host name; or, selecting a different logical port or network name.

The *db2nodes.cfg* file contains one line for each database partition that belongs to an instance. Each line has the following format:

*dbpartitionnum hostname [logical-port [netname]]*

Tokens are delimited by blanks. The variables are:

- ▶ **dbpartitionnum**: The database partition number, which can be from 0 to 999, uniquely defines a database partition. Database partition numbers must be in an ascending sequence. You can have gaps in the sequence. Once a database partition number has been assigned, it cannot be changed. Otherwise the information in the distribution map, which specifies how data is distributed, will be compromised. If you drop a database partition, its database partition number can be used again for any new database partition that you add. The database partition number is used to generate a database partition name in the database directory.

It has the format: `NODEnnnn`.

Here “nnnn” is the database partition number, which is left-padded with zeros. This database partition number is also used by the `CREATE DATABASE` and `DROP DATABASE` commands.

- ▶ **hostname**: The host name of the IP address for inter-partition communications. Use the fully-qualified name for the hostname. The



/etc/hosts file should also use the fully-qualified name. If the fully-qualified name is not used in the db2nodes.cfg file and in the /etc/hosts file, you might receive an error message SQL30082N RC=3. The exception to this is when netname is specified. In this situation, netname is used for most communications, with the hostname only being used for db2start, db2stop, and db2\_all.

- logical-port: This parameter is optional, and specifies the logical port number for the database partition. This number is used with the database manager instance name to identify a TCP/IP service name entry in the etc/services file. The combination of the IP address and the logical-port is used as a well-known address, and must be unique among all applications to support communications connections between database partitions. For each host name, one logical-port must be either 0 (zero) or blank (which defaults to 0). The database partition associated with this logical-port is the default node on the host to which clients connect. You can override this with the DB2NODE environment variable in db2profile script, or with the sqlesetc() API.
- netname: This parameter is optional, and is used to support a host that has more than one active TCP/IP interface, each with its own hostname.

### 12.1.3 Catalog partition

The SYSCATSPACE table space contains all the DB2 system catalog information (metadata) about the database. In a DPF environment, SYSCATSPACE cannot be partitioned, but must reside in one partition, known as the catalog partition. When creating a database, the partition on which the Create Database command is issued becomes the Catalog partition for the new database. All access to system tables goes through this database partition.

### 12.1.4 Partition groups

You are not restricted to having all table spaces divided across all database partitions in the database. You can divide table spaces across a subset of database partitions in the system. A database partition group is a logical layer that allows the grouping of one or more partitions to perform operations on all the partitions in the group. A database partition can belong to more than one partition group. The following are examples of creating a database partition:

```
CREATE DATABASE PARTITION GROUP GROUPA ON DBPARTITIONNUMS (1,2,4)
CREATE DATABASE PARTITION GROUP GROUPB ON DBPARTITIONNUMS (3)
CREATE DATABASE PARTITION GROUP GROUPC ON DBPARTITIONNUMS (1,2,3,4)
```

DB2 DPF table spaces are created in database partition groups. The CREATE TABLESPACE statement with the IN DATABASE PARTITION GROUP clause can be used for this purpose. Tables are then created in the table spaces. When

a table is created in a multi-partition group table space, some of its rows are stored in one partition, and other rows are stored in other partitions. Usually, a single database partition exists on each physical node, and the processors on each system are used by the database manager at each database partition to manage its own part of the total data in the database. Figure 12-4 illustrates a database with four partitions and three partition groups.

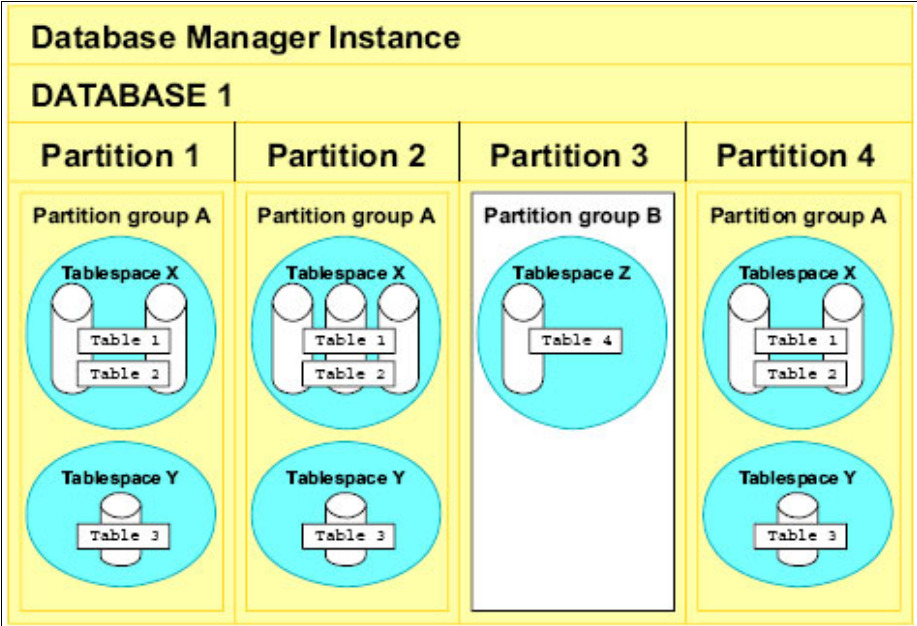


Figure 12-4 Database partition groups

### 12.1.5 Distribution keys and distribution maps

Users can choose how to partition their data by specifying distribution keys in the create table statement or using the default distribution key. It is very important to choose an appropriate partition key as this determines the partition where each row of data is stored and has a big impact on performance. Data should be partitioned to avoid cross-partition joins. Ideally we want all data referred to by all statements in any given transaction to reside in one node (collocation).

When a database partition group is created, a distribution map<sup>2</sup> is generated. A distribution map is just an array, or a vector, of 4096 entries, each of which maps to one of the database partitions of the database partition group. DB2 uses a hashing algorithm to partition the rows. A distribution key value is hashed to generate the partition map index value. The hash value ranges from 0 to 4095.

<sup>2</sup> Prior to DB2 9, distribution map was referred as partitioning map.

The distribution map entry for the index provides the database partition number for the hashed row.

Figure 12-5 shows the partition map for database partition group 1 with partitions 1,2,4.

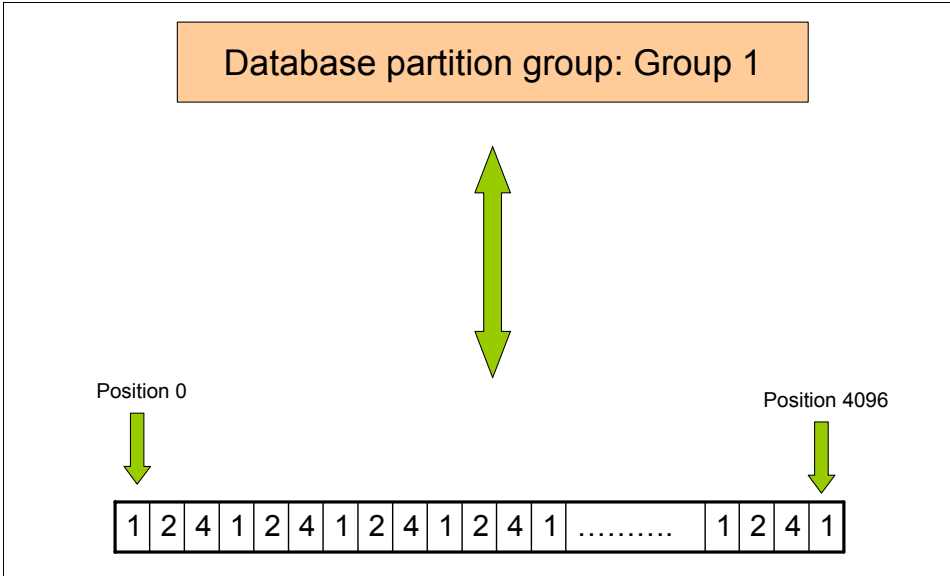


Figure 12-5 Partition Map for Database Partition Group 1

Figure 12-6 shows how DB2 determines which partition a given row belongs to. Here the partition key EMPNO (value 120) is hashed to a value 9 which is used to index to the partition number 1.

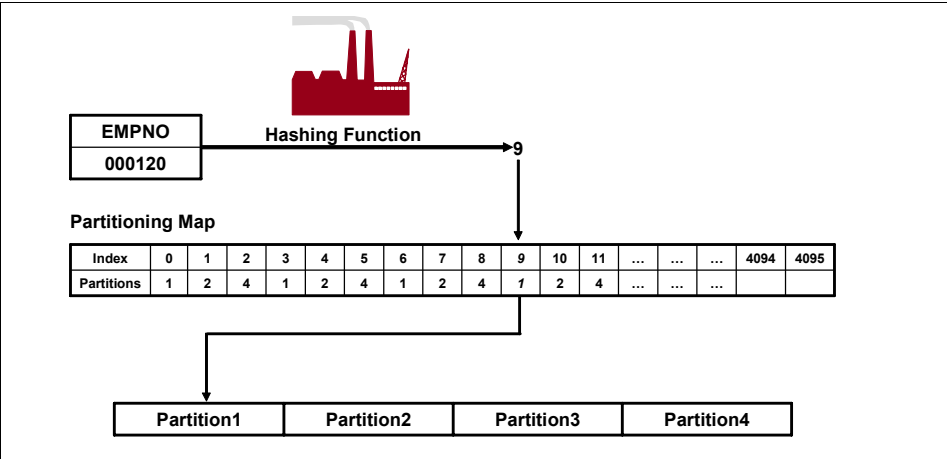


Figure 12-6 DB2 process to identify the partition the row has to be placed

In certain cases choosing an appropriate partition key may not be possible and can cause data skew. *Data skew* is a situation where data is not evenly distributed across the nodes, that is, most of the data ends up on one of the nodes. If that happens, you can no longer ensure performance with scalability, and can only run as fast as the slowest node on the system. In this situation, nodes with heavy data concentration will become a bottleneck. DB2 has updatable partition maps to allow you to redistribute the data and correct the data skew.

## 12.2 Installation

There is no technical prerequisite to enable DPF on existing DB2 ESE servers other than purchasing the required license and following a configuration checklist.

### 12.2.1 High level installation procedure

Figure 12-7 illustrates a conceptual DB2 partitioned database environment. We describe a high level installation procedure using this environment on AIX.

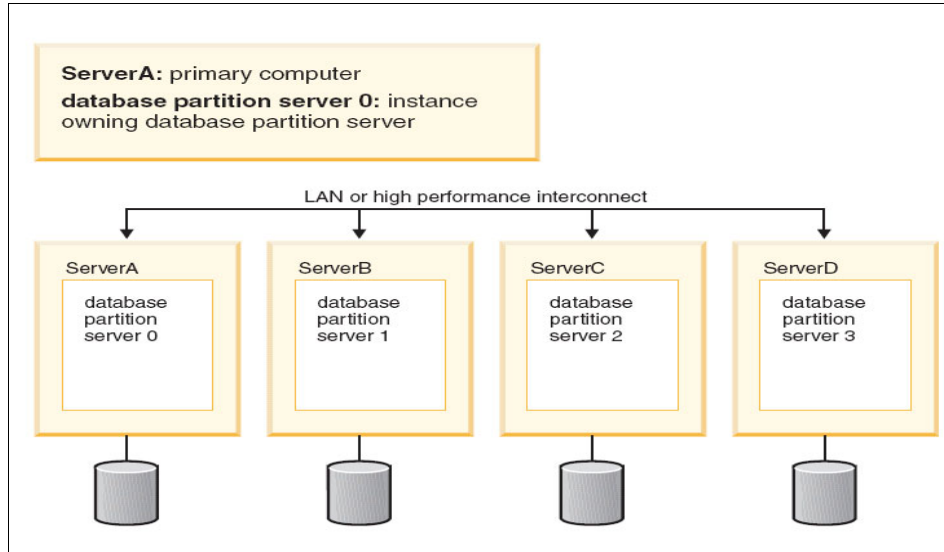


Figure 12-7 Server A is the instance owning partition server as it is the primary computer

Each of the installation steps to configure DPF on an AIX cluster is described in detail in *Quick Beginnings for DB2 Servers*, GC10-4246.

The following lists the steps to configure DPF on an AIX cluster:

1. Verify that each computer meets the necessary operating system, memory and disk requirements. The minimum requirements for hardware, operating system or distribution, software, communication, DB2 Administration Server (DAS), memory, and disk for the installation of a partitioned DB2 server must be met.
2. Update kernel parameters on each computer (HP-UX, Linux, Solaris). Check the manual for recommended kernel settings and how they can be changed for HP-UX, Linux, and Solaris.
3. Modify environment settings on each computer (AIX). Use the manual to guide you on the recommended settings for the AIX *maxuproc* device attribute, as well as TCP/IP network parameters. The manual also provides recommendations on paging space and the number of network file system daemons (NFSDs) that should be running on every computer.
4. Verify that NFS daemon is running on the computer. Create a DB2 home file system on the primary computer and share it with participating computers. Create a DB2 home file system on the primary computer, NFS export the home file system, and NFS mount the home file system on each participating computer.

5. Create required users and groups on each computer. Three users and groups are generally required to operate DB2: an Instance owner and group, fenced user and group, and DB2 Administration Server user and group.

- a. Create a group for the DB2 instance owner (for example, db2iadm1), the fenced group that will execute user defined functions (UDFs) or stored procedures (for example, db2fadm1), and the Administration Server group (for example, db2asgrp) by entering the following commands:

```
mkgroup id=999 db2iadm1
mkgroup id=998 db2fadm1
mkgroup id=997 db2asgrp
```

- b. Create a user that belongs to each group that you created in the previous step using the following commands. The home directory for each user will be the DB2 home directory that you previously created and shared (/db2home).

```
mkuser id=1004 pgrp=db2iadm1 groups=db2iadm1
home=/db2home/db2inst1 core=-1 data=491519 stack=32767 rss=-1
fsize=-1 db2inst1
```

```
mkuser id=1003 pgrp=db2fadm1 groups=db2fadm1
home=/db2home/db2fenc1 db2fenc1
```

```
mkuser id=1002 pgrp=db2asgrp groups=db2asgrp home=/db2home/db2as
db2as
```

- c. Set an initial password for each user that you created.
- d. Create the same user and group accounts on each computer that will participate in your partitioned database system. For our example, perform this task on ServerB, ServerC, and ServerD (Figure 12.7) and log in to each, ensuring the passwords are set. Users and groups must be identical on all participating computers in the cluster (exactly the same UIDs and GIDs).

If an existing user is used as the Administration server user, this user must also exist on all the participating computers before installation. If you use the DB2 Setup wizard to create a new user for the Administration server on the instance owning computer, then this user will also be created (if necessary) during the response file installations on the participating computers. If the user already exists on the participating computers, it must have the same primary group

6. After preparing your environment, install DB2 on each computer locally. You must install the instance-owning partition first (ServerA) and create a response file that is used to install on the other computers (ServerB, ServerC, ServerD). This ensures that the same components are installed and

configured the same way. You should also create a local administration contact list on the instance-owning partition (ServerA). When the DB2 Administration Server is installed and configured on the other participating computers, it will be configured to use the contact list on the instance owning computer.

- a. You must have already created the users and groups (as outlined above) on all the nodes where DB2 is to be installed. Do not let the db2setup program define the users and groups. If this happens, there is a possibility that the same user (db2inst1, for instance) might have different User IDs on the various nodes, which leads to ownership and execution problems on the shared /db2home.
  - b. When installing through the DB2 Setup Wizard, on the Set up a DB2 instance panel, select **Create a DB2 instance**. On the Select how this instance will be used panel, select **Partitioned** instance.
7. Transfer the response file to the other servers and run an unattended installation using the command **db2setup -r <response file>** on each server. Check the messages in the log file when the installation finishes. The location of the log file is /tmp/db2setup.log on AIX.
  8. Apply the latest DB2 FixPak on all the participating computers in the DPF configuration. When installing a FixPak on a DPF system, all participating computers must have the same FixPak installed.
  9. The instance owner should update the node configuration file db2nodes.cfg. The information you provide in the node configuration file tells DB2 which database partition servers will participate in the instance. There is a db2nodes.cfg file for each instance in a partitioned database environment. The db2nodes.cfg file must contain one entry for each database partition server that will participate in the instance.

The db2nodes.cfg file is locked when the instance is running and can only be edited when the instance is stopped. After you have finished updating the db2nodes.cfg file restart the instance

When you create an instance, the db2nodes.cfg file is automatically created and an entry for the instance-owning database partition server is added. For example, when you created the DB2 instance using the DB2 Setup wizard, on the primary computer ServerA, the db2nodes.cfg file looks like the following:

```
0 ServerA 0
```

This entry includes the database partition server number (node number), the TCP/IP host name of the server where the database partition server resides, and a logical port number for the database server partition.

If you are installing a DPF configuration as shown in Figure 12-7 on page 367, with four computers and a database partition server on each computer, the updated db2nodes.cfg should appear similar to the following:

```
0 ServerA 0
1 ServerB 0
2 ServerC 0
3 ServerD 0
```

If you are not using a clustered environment and want to have four database partition servers on one physical workstation called ServerA, update the db2nodes.cfg file as follows:

```
0 ServerA 0
1 ServerA 1
2 ServerA 2
3 ServerA 3
```

10. The db2nodes.cfg file is also used to specify the IP address or host name of a high-speed interconnect, should you want to use a high-speed interconnect for database partition server communication. If you have a DPF configuration with four computers, four database partition servers with high speed switches, update the db2nodes.cfg file as follows:

```
0 ServerA 0 Switch1
1 ServerB 0 Switch2
2 ServerC 0 Switch3
3 ServerD 0 Switch4
```

11. Enable communication between database partition servers that participate in your partitioned database system. Communication between database partition servers is handled by the Fast Communications Manager (FCM). To enable FCM, a port or port range must be reserved in the /etc/services file on each computer in your partitioned database system. During instance creation a number of ports equal to the number of logical nodes that the instance is capable of supporting will be reserved in the /etc/services. The Fast Communication Manager will use these ports. The reserved ports will be in the following format (port numbers may vary):

```
DB2_InstanceName      60000/tcp
DB2_InstanceName_1    60001/tcp
DB2_InstanceName_2    60002/tcp
DB2_InstanceName_END  60003/tcp
```

The only mandatory entries are the beginning (DB2\_InstanceName) and ending (DB2\_InstanceName\_END) ports. The other entries are reserved in the services file so that other applications do not use these ports.

The services file must be updated with the correct number of entries depending on the number of partitions being configured for that DPF



instance. When partitions within the same server are involved, communication between the partitions still requires this setup.

12. Enable execution of remote commands (completed by the instance owner). In a partitioned database system, each database partition server must have the authority to perform remote commands on all the other database partition servers participating in an instance. This can be done by updating the `.rhosts` file in the home directory for the instance. Because the home directory for the instance is on the shared DB2 home file system, only one `.rhosts` file is required.

Add entries to the `.rhosts` file for each computer including the primary computer. The `.rhosts` file has the following format:

```
hostname instance_owner_user_name
```

Some systems may require a long host name to be specified, for example:

```
ServerA.yourdomain.com
```

Before you add host name entries to the `.rhosts` file, make sure the host names in the `/etc/hosts` and the `/etc/resolv.conf` files can be resolved.

The `$INSTHOME/.rhosts` file should contain entries similar to the following for a DB2 Instance `db2inst1`:

```
ServerA.yourdomain.com db2inst1
ServerB.yourdomain.com db2inst1
ServerC.yourdomain.com db2inst1
```

Instead of specifying each host name individually, you may specify the following entry in the `.rhosts` file, but this may pose a security risk and should only be done in a test environment.

```
+ db2inst1
```

If you have specified a high-speed switch (`netname`) in the `db2nodes.cfg` file, you must also add `netname` entries for each computer to the `.rhosts` file. The `netname` values are specified in the fourth column of the `db2nodes.cfg` file. A `.rhosts` file with high-speed switch (`netname`) entries may look similar to the following:

```
ServerA.yourdomain.com db2inst1
ServerB.yourdomain.com db2inst1
ServerC.yourdomain.com db2inst1
ServerD.yourdomain.com db2inst1
Switch1.yourdomain.com db2inst1
Switch2.yourdomain.com db2inst1
Switch3.yourdomain.com db2inst1
Switch4.yourdomain.com db2inst1
```

13. Enable Control Center administration. Before you can use the Control Center to administer your partitioned database system, you must start the DB2 Administration server on all computers.

In this example, db2as is the DAS user. Enter the following command to start the DB2 Administration Server:

```
$DASHOME/das/bin/db2admin start
```

Where DASHOME is the home directory for the DB2 Administration Server.

## 12.2.2 Verifying the installation

The following procedure can be used to verify the installation:

1. Log on to the primary computer (ServerA) as the instance-owning user. In our installation example, db2inst1 is the instance-owning user. Start the database manager by entering the **db2start** command.
2. Before creating the database, you must check the value of database manager configuration parameter DFTDBPATH. This is the path where the database will be created by default. The instance owner's home directory is the default database path. It is recommended to not create the database on the instance home directory which is shared by all the partitions. Create a path locally in all the individual partitions and create the database on that path, so that the data is spread across the different partition servers. The local path has the same name on all the partitions. Enter the **db2sample** command to create the SAMPLE database with this local path name. Run this command:

```
db2sample <localpath>
```

The SAMPLE database is automatically cataloged with the database alias SAMPLE when it is created.

3. Verify the partition currently connected using the following command:

```
db2 "values (current dbpartitionnum)"
```

You can switch between partitions in your DPF environment by using the DB2NODE environment variable. For example to make partition 0 the active partition.

```
DB2NODE=0  
export DB2NODE
```

4. Enter the following DB2 commands from a DB2 command window to connect to the SAMPLE database, and retrieve a list of all the employees that work in department 20:

```
db2 CONNECT TO sample  
db2 "SELECT * FROM staff WHERE dept = 20"
```

5. To verify that data has been distributed across database partition servers, enter the following commands from a DB2 command window:

```
SELECT DISTINCT DBPARTITIONNUM(empno) FROM employee;
```

The output will list the database partitions used by the EMPLOYEE table. The specific output will depend on the number of partitions in the database and the number of partitions in the partition group used by the table space where the EMPLOYEE table was created.

## 12.3 Scaling the database

As the customer data volumes grow, and more users are added, a scalable database must be able to deliver good performance and keep pace with the growth. By designing a multi-partition database properly you can expect the DB2 DPF system to scale in a near-linear fashion. Because the number of database partitions has little impact on the inter-partition traffic, DB2 has a very scalable performance. For achieving good scalability it is important to understand the concept of *balanced configuration unit* (BCU).

### 12.3.1 Balanced configuration unit

Scalability is achieved by splitting the data across multiple DB2 database partitions. These database partitions reside on a stack of technology which includes the DB2 software, operating system, processors, memory, I/O backplane or I/O paths, disk controllers and individual disk spindles. A balanced configuration unit is a subset of the whole database that is designed to meet the customer requirements for performance, failover, and support. Multiple BCUs will be combined to complete the entire data warehouse design and to fulfill customer requirements. You may think of the BCU as a building block. Multiple identical building blocks are combined to form the entire database solution. See Figure 12-8.

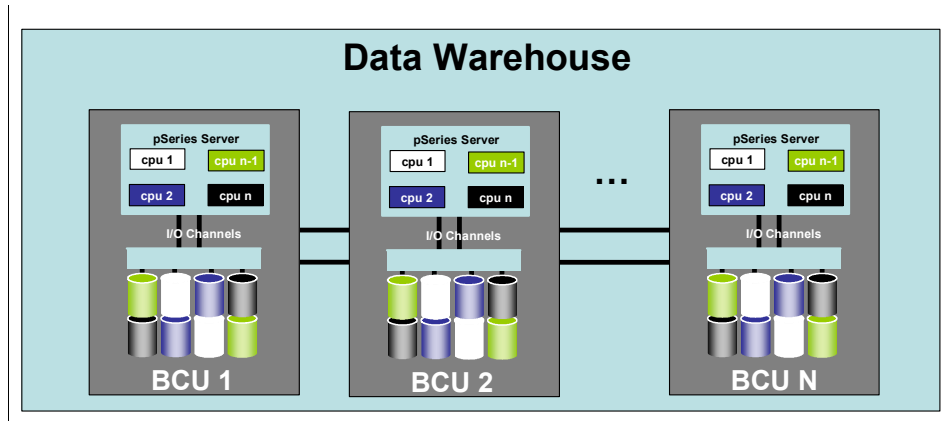


Figure 12-8 *Balanced configuration unit*

A BCU is the minimum replicable hardware/software stack necessary to start or expand the partitioned database system to accommodate the growing business data requirements. A BCU is a single node with a scalable ratio from disks to I/O to memory to CPUs to network. The *balanced partition unit* (BPU) is the smallest unit of infrastructure that can be formed. A BPU is a DB2 database partition with multi-partition data and server “components”. A BCU can be composed of multiple BPUs. Therefore, if one understands the behavior of one partition, one can understand the behavior of the BCU, and thus of the entire infrastructure.

### 12.3.2 Scaling the number of partitions

Data growth can lead to imbalance of CPU/disk and other resources. We must identify the new number of BCUs to implement the solution which can handle the increase in data growth. Once we have identified the required number of BCUs

and BPU's we can add the appropriate number of DB2 partitions. As illustrated in Figure 12-9, once the new partitions are added, the data can be redistributed among the database partitions to achieve a balanced database partition group.

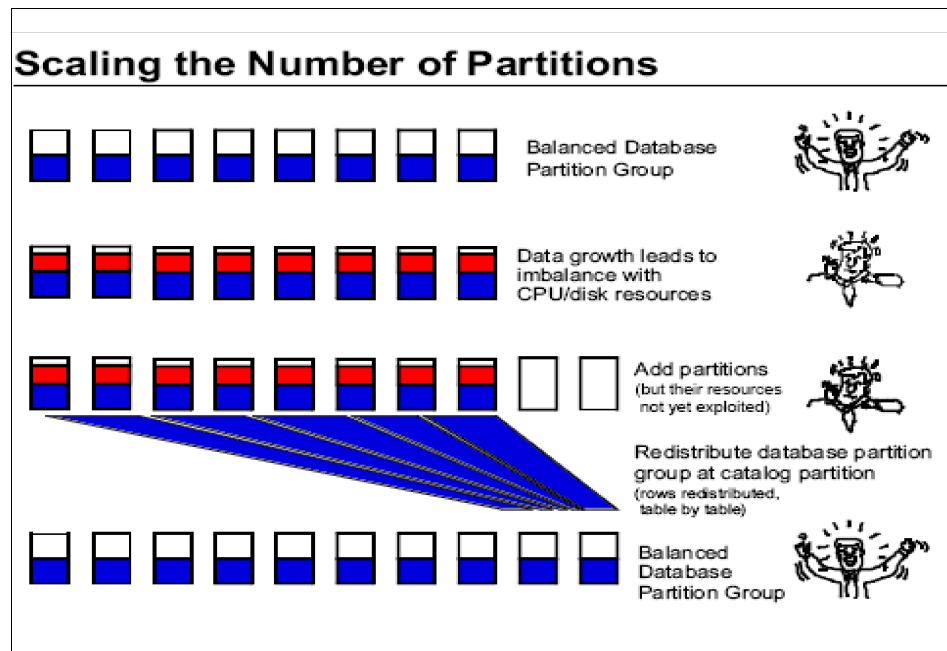


Figure 12-9 Scaling the number of partitions

DB2 provides commands to scale your configuration by adding a new database partition to your existing partitioned database system. Figure 12-10 illustrates scaling out the database by adding a partition on a new node.

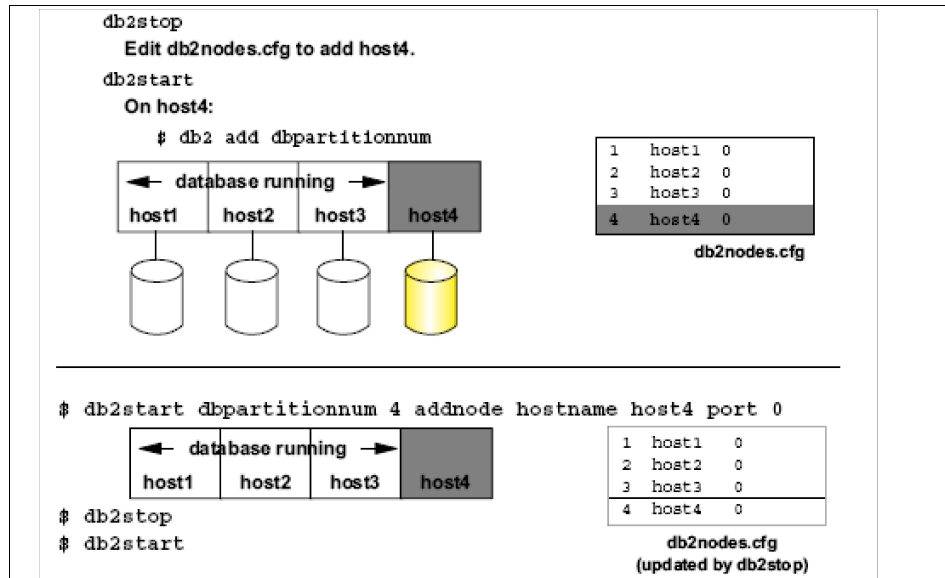


Figure 12-10 Different methods for Adding a DB2 partition

When partitions are added to a database, the data could be rebalanced to take advantage of the new database partitions. The *redistribute database partition group* utility moves data among database partitions, after adding a database partition, before removing a database partition or during load balancing, or both. It must be executed from catalog partition. An example of how to use the redistribute command when we add a partition is given below:

1. Update the database partition group definition.  

```
ALTER DATABASE PARTITION GROUP GRP1 ADD DBPARTITIONNUM(5) WITHOUT TABLESPACES
```
2. Extend all the table spaces (defined in this database partition group) to the new partition with partition-specific container definitions.  

```
ALTER TABLESPACE TBS1 ADD (FILE '/DATA/TB' 2048) ON DBPARTITIONNUM(5)
```
3. Redistribute the data uniformly for all the tables created in the table spaces in the database partition group GRP1 across the newly added partition and update the GRP1's partition map.  

```
REDISTRIBUTE DATABASE PARTITION GROUP GRP1 UNIFORM
```

DB2 also provides commands to drop a database partition.

Before issuing a drop, if data exists on the database partition being dropped, you must redistribute the data that resides on this database partition for every database to ensure that data is not lost. The following are the steps to drop a partition

1. Redistribute the data that resides on the database partition.
2. Issue the **drop dbpartitionnum verify** command on the database partition to be dropped to verify that the database partition is not in use.
  - If the SQL6034W message is received, you can proceed with the drop partition process.
  - If the SQL6035W message is received, you must use the **redistribute database partitiongroup** command to move the data from the database partition being dropped to other database partitions of the database. You cannot drop the database partition until this is completed.
3. Issue the **db2stop drop dbpartitionnum** command.
4. Restart the database with **db2start**.

## 12.4 High availability on DPF

Data availability at the processor level is an issue if a node goes down in a shared nothing architecture. In this case, the data on the node is not available until the administrator recovers the node. In a shared disk, you can lose a processor, and still access a disk from another processor. In a DPF shared nothing environment, if you lose a processor, you will no longer have access to that data.

HADR is not supported on DB2 ESE with DPF. For high availability on DPF, DB2 provides failover support on many platforms by using platform-specific high availability cluster software:

- ▶ On AIX, DB2 supports HA failover through the capabilities of High Availability Cluster Multi-Processing, Enhanced Scalability (HACMP ES).
- ▶ On Windows, you can implement HA failover support with Microsoft Cluster Server (MSCS).
- ▶ On the Solaris Operating Environment, you can implement HA failover support with Sun Cluster or VERITAS Cluster Server.
- ▶ On Hewlett-Packard, you can implement failover support with Multi-Computer/ServiceGuard.
- ▶ On Linux, Tivoli System Automation (TSA) Base Component provides first-tier clustering/failover support.

Figure 12-11 illustrates an HACMP solution for a highly available DB2 DPF system.

For detailed information about HA failover support and configuration, refer to *Data Recovery and High Availability Guide and Reference*, SC10-4228, or see Chapter 10, “HADR with clustering software” on page 253.

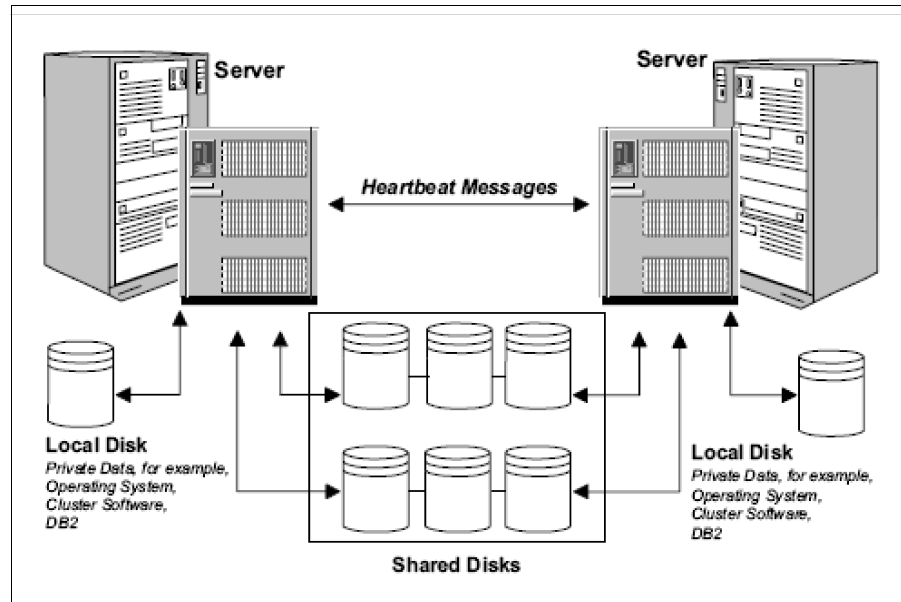


Figure 12-11 HACMP cluster solution

High availability in a DPF environment requires a cluster of systems. A cluster is a group of connected systems that work together as a single system. Clustering allows servers to back up each other when failures occur, by picking up the workload of the failed server.

Normally, DPF is a shared-nothing environment, but in an HA environment it can be configured such that disks are shared in support of failover. Two systems in a cluster can each have an independent database workload which access the data on non-shared disks. In the event of a failover, this data will not get recovered on the other server. Only the data that utilizes the shared disks is eligible to be recovered on the other backup server. What actually happens in the event of a failover is controlled by HA software scripts, which can be easily tailored to your unique environment.

In a DB2 DPF environment, when a partition fails on a node in a cluster, the HA software detects the failed partition and restarts the partition on another node. The HA failover script updates the `db2nodes.cfg` file and restarts the failed



partition on the backup server/node. If a server with multiple DB2 partitions fails, it is important that the failover script starts the failed logical DB2 partitions on the backup server in the order that is defined in the db2nodes.cfg file. The logical database partition with port number 0 must always be started first.

In a DPF environment, the HA clustering software can detect a failed node and can restart the lost node on an empty spare, in which case the data load stays balanced, or the lost node can be restarted on another node that already has data, in which case the load is unbalanced because the takeover node is working harder than the others.

The three most common failover strategies are known as *Idle Standby*, *Mutual Takeover* and *Hot/Active Standby*. See Figure 12-12.

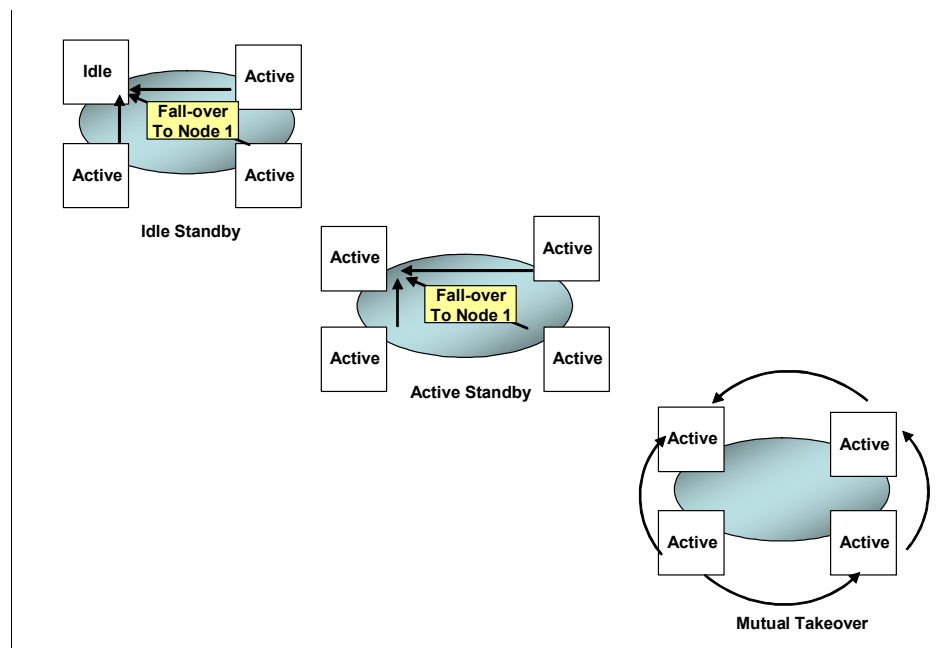


Figure 12-12 HA strategies for DPF systems

- ▶ **Idle Standby:** In this configuration, one system is idle, or in standby mode, ready to take over the instance if there is an operating system or hardware failure involving any of the other systems. Overall system performance is not impacted, because the standby system is idle until needed.
- ▶ **Active Standby:** Another implementation is where one server (a Hot Standby) is prepared to takeover any of the other servers in the group which fails. All the servers in this DPF configuration are active. You must also configure a backup for the hot standby.

- **Mutual Takeover:** In this configuration, each system is the designated backup for another system. In the Active Standby and Mutual Takeover configurations the overall system performance may be impacted, because the backup system must do extra work following a failover. It must do its own work plus the work that was being done by the failed system.

Figure 12-13 shows an example of Hot Standby Takeover where Server 4 is configured to be the backup of Server 1 which is running partition 1. Both the servers, Server1 and Server4 are in active mode running partitions 1 and 4. When Server 1, which is running DB2 partition 1, goes down, the HA software fails over partition 1 to the hot Server4 which is already running partition 4.

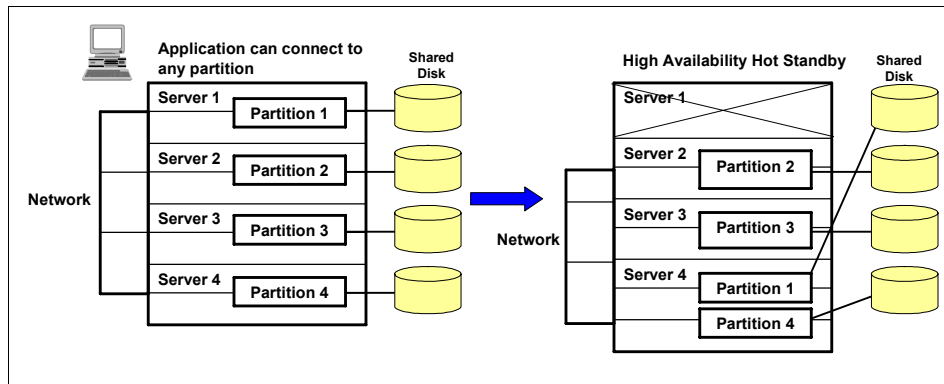


Figure 12-13 Partition 1 is failed over from server 1 to Hot standby server 4

The failover script updates the db2nodes.cfg and restarts partition 1 on Server4. Example 12-1 shows a portion of the db2nodes.cfg file before and after the failover.

#### Example 12-1 db2nodes.cfg

Before:

```
1 Server1 0
2 Server2 0
3 Server3 0
4 Server4 0
```

```
db2start dbpartitionnum 1 restart hostname Server4 port 1
```

After:

```
1 Server4 1
2 Server2 0
3 Server3 0
4 Server4 0
```

## References

For more information to supplement this chapter, see the following references:

- ▶ *A colorful introduction to DB2 UDB, Version 8 for UNIX, Linux, and Windows: Overview of the Database Partitioning Feature:*  
<http://www.ibm.com/developerworks/db2/library/techarticle/dm-0403chong/>
- ▶ *DB2 Version 9 Administration Guide: Implementation*, SC10-4221.





# Table partitioning

In this chapter we discuss a new feature of large database management: table partitioning, introduced in DB2 9. Table partitioning allows database administrators (DBAs) to divide table data into partitions and store it in one or multiple table spaces. It also provides easier management, improved performance, and greater scalability for large databases.

We introduce the following topics:

- ▶ Concepts of partitions
- ▶ Performance impact of table partitioning
- ▶ How to create and manipulate partitioned tables
- ▶ How to use this new feature, with examples

## 13.1 Introduction

*Table partitioning* feature is a table organization scheme in which table data is divided into partitions according to the values in one or more table partition key columns of the table. Each data partition is stored separately in a storage object called *data partition* or *range*. These storage objects can be in different table spaces, in the same table space, or a combination of both. The decisions on where to place the storage objects and the data range that each storage object will contain are made at the time of creating the table, or by altering the table using the PARTITION BY clause.

DB2 9 supports data partitions or data ranges based on a variety of attributes. A commonly used partitioning scheme is the date, by which you can group data into partitions by year or month. You can also use numeric attributes for partitioning. For instance, records with IDs from 1 to 1million are stored in one data partition, IDs from 1million to 2 million are stored in another data partition, and so on. Another example is that you can have records for customers with names starting from A-C in one data partition, D-M in the second data partition, N-Q in third data partition, and R-Z in the last data partition.

Data partition names or numbers are useful for data partition operations. Although you have the option to reference a data partition by its name or number, they can be completely transparent to applications. That is, applications can access data by specifying the column and table names only, and do not have to worry about which data partition(s) the data resides in.

A partitioned table simplifies the rolling in and rolling out of table data, and can contain vastly more data as compared to an ordinary table. You can create a partitioned table with a maximum of 32767 data partitions. Data partitions can be added to, attached to, and detached from a partitioned table, and you can store multiple data partition ranges from a table in one table space. You can use the ALTER TABLE ... ATTACH statement to incorporate (roll-in) an existing table into a partitioned table. The roll-out operation can be accomplished with the ALTER TABLE ... DETACH statement.

Using partitions can also increase query performance by scanning only the relevant table partitions instead of the entire table, to get a consolidated result. The SQL engine can skip partitions during data read if a partition does not contain the relevant information to resolve the query.

Table data is partitioned as specified in the PARTITION BY clause of the CREATE TABLE statement. The columns used in this definition are referred to as *table partitioning key columns*. This organization scheme can be used in isolation or in combination with other organization schemes. By combining the DISTRIBUTE BY and PARTITION BY clauses of the CREATE TABLE statement, data can be spread across database partitions spanning multiple table spaces. The DB2 organization schemes include:

- ▶ DISTRIBUTE BY HASH
- ▶ PARTITION BY RANGE
- ▶ ORGANIZE BY DIMENSIONS

Partitioned hierarchical or temporary tables, range-clustered tables, and partitioned views are not supported in DB2 V9.1. Also, the following column types are not supported for use in partitioned tables:

- ▶ XML
- ▶ DATALINK

The following columns cannot be used as the partitioning key for a partitioned table:

- ▶ Long Varchar
- ▶ Long Vargraphic
- ▶ Binary Large Object (BLOB)
- ▶ Character Large Object (CLOB)
- ▶ Double-byte character large object (DBCLOB)

## 13.2 Benefits of table partitioning

Table partitioning has many benefits including fast roll-in and roll-out, easier administration of large tables, flexible index placement, and increased query performance. Some of the situations in which table partitioning can be particularly beneficial are as follows:

- ▶ You have a large data warehouse, where you usually add or delete parts of the information based on a known criteria, for example date. In such a situation the system can benefit from easier roll-in and roll-out of table data.
- ▶ You have a data warehouse with large tables and you have a criteria for queries that could split the data among several devices.
- ▶ You are required to use Hierarchical Storage Management (HSM) solutions more effectively.
- ▶ You are required to place indexes in different devices, depending on the speed of the devices and how often the indexes are used.

## 13.2.1 Efficient roll-in and roll-out

Table partitioning provides the capability of rolling data partition into and out from a table without having to take the database offline. By attaching a new partition, you can add an existing table including data, to a partitioned table as a new partition. The recently added partition will conform to certain criteria for the partitioning range. You also can split a table partition as an independent table by detaching the partition.

This fast roll-in and roll-out capability is particularly useful in maintaining a certain range of data online and in archiving the out-dated data or keeping the out-dated data in a historical table. For example, a manufacturing site can have a partitioned table which stores one year of failure analysis data. The table is partitioned by month. At the beginning of each month the oldest fragment of data is detached from the table and attached to the historical table. A new fragment of data is added for the month that is beginning.

## 13.2.2 Easier administration of large tables

Table level administration is more flexible because you can perform administrative tasks on individual data partitions. These tasks include:

- ▶ Detaching and reattaching a data partition
- ▶ Backing up and restoring individual data partitions
- ▶ Reorganizing individual indexes

Time-consuming maintenance operations can be shortened by breaking them into a series of smaller operations. For example, you can backup a partitioned table one data partition at a time if the data partitions are placed in separate table spaces.

## 13.2.3 Flexible index placement

Individual indexes from the same table can now be placed in different table spaces allowing more granular control of index placement. Some benefits of this new design include:

- ▶ Improved performance of drop index and online index create.
- ▶ Ability to use different values for any of the table space characteristics between each index on the table (for example, different page sizes for each index may be appropriate to ensure better space utilization).
- ▶ Reduced input and output (IO) contention providing more efficient concurrent access to the index data for the table.



- ▶ When individual indexes are dropped space becomes immediately available to the system without the need for an index reorganization.
- ▶ If you choose to perform index reorganization, an individual index can be reorganized.

Both database managed space (DMS) and system managed space (SMS) table spaces support the use of indexes in a different location than the table.

If you need the flexibility of placing different indexes in different table spaces but do not really need a partitioned table, you can create a partitioned table consisting of only one partition.

### 13.2.4 Data partition elimination during query processing

Query processing is enhanced to automatically eliminate data partitions based on the predicates of the query. The data partition elimination ability benefits many decision support queries. It also increases the throughput for OLTP systems by allowing the use of multiple devices to insert, update, and delete table data at the same time.

## 13.3 Creating a partitioned table

Partitioned tables can be created by using the CREATE TABLE statement with the PARTITION BY clause or using Control Center. To partition an existing table, you must unload the data, drop the table and recreate it as a partitioned table.

The ALTER TABLE statement allows you to add partitions to a table. The ALTER TABLE statement with DETACH or ATTACH clauses can be used to add an existing table as a partition of a partitioned table or to convert a data partition into an independent table.

### 13.3.1 Prerequisites

To create a table, the privileges held by the authorization ID of the statement must include at least one of the following authorities or privileges:

- ▶ CREATETAB authority on the database and USE privilege on all the table spaces used by the table, as well as one of the following:
  - IMPLICIT\_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist.
  - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema.

- ▶ SYSADM or DBADM authority.

To detach a data partition from a partitioned table, the user must have the following authorities or privileges:

- ▶ The user performing the DETACH operation must have the authority required to ALTER, SELECT and DELETE from the source table.
- ▶ The user must also have the authority required to create the target table. Therefore, to alter a table in order to detach a data partition, the privileges held by the authorization ID of the statement must include at least one of the following authorities or privileges on the target table:
  - SYSADM or DBADM authority
  - CREATETAB authority on the database and USE privilege on the table spaces used by the table, and also one of:
    - IMPLICIT\_SCHEMA authority on the database, if the implicit or explicit schema name of the table does not exist.
    - CREATEIN privilege on the schema, if the schema name of the table refers to an existing schema.

To attach a data partition to a partitioned table, the user must have the following authorities or privileges:

- ▶ The user performing the attach must have the authority needed to ALTER and to INSERT into the target table.
- ▶ The user must also be able to SELECT from and to DROP the source table. Therefore, to alter a table in order to attach a data partition, the privileges held by the authorization ID of the statement must include at least one of the following on the source table:
  - SELECT privilege on the source table and DROPIN privilege on the schema of the source table.
  - CONTROL privilege on the source table.
  - SYSADM or DBADM authority.

### 13.3.2 Creating partitioned table examples

Example 13-1 shows how to create a partitioned table named CUSTOMER where rows will be distributed across four partitions. Each partition will be placed in one of the four table spaces ts1, ts2, ts3, and ts4. Rows will be placed in each partition based on the value in column l\_shipdate.

*Example 13-1 Creating partitioned table CUSTOMER*

---

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
IN TBLSPC_01,TBLSPC_02,TBLSPC_03,TBLSPC_04
PARTITION BY RANGE(l_shipdate)
(
    STARTING FROM ('01/01/2006') ENDING AT ('12/31/2006') EVERY (3
    MONTHS)
)
```

---

The rows whose l\_shipdate value is higher or equal to '01/01/2006' but lower than '04/01/2006' will be stored in the partition in ts1. If l\_shipdate value is higher or equal to '04/01/2006' but lower than '06/01/2006', the record will be stored in the partition in ts2 and so on.

If you include more table spaces than the ranges, you will receive a warning message stating that the extra table spaces are ignored. In Example 13-2 we change the range partitioning so that only three table spaces are needed. However, the IN clause still specifies four table spaces. As you can see, the table is created but SQL20302W is returned.

*Example 13-2 Creating partitioned table with extra table space specified*

---

```
$ db2 -tvf tab3.sql
drop table customer
DB20000I The SQL command completed successfully.

CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30)) IN
TBLSPC_01,TBLSPC_02,TBLSPC_03,TBLSPC_04 PARTITION BY RANGE(l_shipdate)
(
    STARTING FROM ('01/01/2006') ENDING AT ('09/30/2006') EVERY (3 MONTHS)
)
SQL20302W More table spaces than required were specified in the IN or
LONG IN clause. The extra table spaces are ignored. SQLSTATE=01675
```

---

Example 13-3 shows an alternate form of the CREATE TABLE statement in which the ranges are implicitly included.

*Example 13-3 Creating partitioned table with range implicitly included*

---

```
CREATE TABLE customer (l_shipdate DATE, l_name CHAR(30))
PARTITION BY RANGE(l_shipdate)
(
  STARTING FROM ('01/01/2006') ENDING AT ('03/31/2006') in TBLSPC_01,
  STARTING FROM ('04/01/2006') ENDING AT ('06/30/2006') in TBLSPC_02,
  STARTING FROM ('07/01/2006') ENDING AT ('09/30/2006') in TBLSPC_03,
  STARTING FROM ('10/01/2006') ENDING AT ('12/31/2006') in TBLSPC_04
)
```

---

The query in Example 13-4 selects all the items that were shipped on April 25, 2006. Since the DB2 engine knows that this data is in partition number 2, only this partition is scanned, reducing the I/O, and increasing performance and availability for other applications.

*Example 13-4 Query data in partitioned table*

---

```
SELECT * FROM customer WHERE l_shipdate = '04/25/2006'
```

---

## 13.4 Rotating data in a partitioned table

You can rotate data in a partitioned table from the DB2 Control Center or from the DB2 command line processor (CLP).

Example 13-5 shows how to create a partitioned table, STOCK, that is partitioned by stock creation date (DATE\_CREATED). Initially, this table contains data only for the year 2001.

*Example 13-5 Table STOCK creation*

---

```
CREATE TABLE stock (stock_ex smallint, stock_name char(16), description
char(128), date_created DATE)
in
TBLSPC_01,TBLSPC_02,TBLSPC_03,TBLSPC_04,
TBLSPC_05,TBLSPC_06,TBLSPC_07,TBLSPC_08,
TBLSPC_09,TBLSPC_10,TBLSPC_11,TBLSPC_12
PARTITION BY RANGE(date_created)
(
  STARTING FROM ('01/01/2001')
  ENDING AT ('12/31/2001')
  EVERY (1 MONTHS)
)
```

---

To use the DB2 Control Center to rotate data in a partitioned table, complete the following steps:

1. Expand the Table folder. The table objects are displayed in the contents pane of the DB2 Control Center window.
2. Right-click the table that you want to alter, and select **Open Data Partitions** from the list of actions.
3. In the Open Data Partitions window, click the button associated with your task. If you are attaching, the Attach Data Partition window opens. If you are detaching, the Detach Data Partition window opens.
4. Specify the required fields.

To use the DB2 commands to rotate data in a partitioned table, execute the ALTER TABLE statement. In the following we demonstrate how to update the stock table by removing the data from December 2001 and replacing it with the latest data from December 2003:

1. Remove the old data from table STOCK.

```
ALTER TABLE stock DETACH PARTITION Dec01 into newtable
```

2. Load the new data. Use LOAD with the REPLACE option to overwrite existing data.

```
LOAD FROM data_file OF DEL REPLACE INTO newtable
```

If there are detached dependents, then you must run the SET INTEGRITY statement on the detached dependents before you can load the detached table.

3. If desired, you can perform data cleansing. Data cleansing activities include:
  - a. Filling in missing values
  - b. Deleting inconsistent and incomplete data
  - c. Removing redundant data arriving from multiple sources
  - d. Transforming data
    - Normalization:  
Data from different sources that represents the same value in different ways must be reconciled as part of rolling the data into the warehouse.
    - Aggregation:  
Raw data that is too detailed to store in the warehouse must be pre-aggregated during roll-in.
4. Attach the new data as a new range.

```
ALTER TABLE stock ATTACH PARTITION dec03  
STARTING '12/01/2003' ENDING '12/31/2003'  
FROM newtable;
```

5. Use the SET INTEGRITY statement to update indexes and other dependent objects. Read and write access is permitted during the execution of the SET INTEGRITY statement.

```
SET INTEGRITY FOR stock ALLOW WRITE ACCESS  
IMMEDIATE CHECKED FOR EXCEPTION IN stock USE stock_ex
```

## 13.5 Locking behavior on partitioned tables

In addition to an overall table lock, there is a lock for each data partition in a partitioned table. This allows for finer granularity and increased concurrency compared to a non-partitioned table. The new data partition lock is identified in the output of the **db2pd** command, event monitors, administrative views, and table functions.

When accessing a table, locking behavior obtains the table lock first, and then acquires data partition locks as dictated by the data accessed. Access methods and isolation levels might require locking of data partitions not in the result set. Once these data partition locks are acquired, they can be held as long as the table lock. For example, a cursor stability (CS) scan over an index might keep the locks on previously accessed data partitions to reduce the costs of re-acquiring the data partition lock if that data partition is referenced in subsequent keys. The data partition lock also carries the cost of ensuring access to the table spaces. For non-partitioned tables, table space access is handled by the table lock. Therefore, data partition locking occurs even if there is an exclusive or share lock at the table level for a partitioned table.

Finer granularity allows one transaction to have exclusive access to a given data partition and avoids row locking while other transactions are able to access other data partitions. This behavior can be a result of the plan chosen for a mass update or due to escalation of locks to the data partition level. The table lock for many access methods is normally an intent lock, even if the data partitions are locked in share or exclusive. This allows for increased concurrency. However, if non-intent locks are required at the data partition level and the plan indicates that all data partitions may be accessed, then a non-intent might be chosen at the table level to prevent deadlocks between data partition locks from concurrent transactions.

## **Locking for SQL LOCK TABLE statements**

For partitioned tables, the only lock acquired for the LOCK TABLE statement is at the table level. No data partition locks are acquired.

This ensures that there is no row locking to the table for subsequent DML statements, and also avoids deadlocks at the row, block, or data partition level. LOCK TABLE IN EXCLUSIVE MODE can also be used to guarantee exclusive access when updating indexes, which is useful in limiting the growth of type 2 indexes during a large update.

## **Row and block lock escalation**

For partitioned tables, the unit of escalation of row and block locks is to the data partition level. This again means that a table is more accessible to other transactions even if a data partition is escalated to share, exclusive, or super exclusive, leaving other non-escalated data partitions unaffected. The transaction might continue to row lock on other data partitions after escalation for a given data partition.

The notification log message for escalations includes the data partition escalated as well as the partitioned table's name. Therefore, exclusive access to an index cannot be ensured by lock escalation. Either the statement must use an exclusive table level lock, an explicit LOCK TABLE IN EXCLUSIVE MODE statement must be issued, or the table must use the LOCKSIZE TABLE attribute. The overall table lock for the access method is chosen by the optimizer, and depends upon data partition elimination. A large update to a table might choose an exclusive table lock if there is no data partition elimination occurring.

## **LOCKSIZE TABLE option in the ALTER TABLE statement**

The ALTER TABLE statement has an option for setting LOCKSIZE TABLE, which ensures that the table is locked in shared or exclusive mode with no intent locks. For a partitioned table this lock strategy is applied to both the table lock and to the data partition locks for any data partitions accessed.

## **Rotating data in a partitioned table**

Rotating data in a DB2 Database for Linux, UNIX, and Windows refers to a method of reusing space in a data partition by removing obsolete data from the table then adding new data. The Table partitioning function allows you to detach the data partition with the obsolete data, then attach a new data partition with the latest data.

## 13.6 Understanding index behavior

Indexes on partitioned tables are similar to indexes for ordinary tables, that is, each index contains pointers to rows in all the data partitions of the table. One important difference, however, is that each index on a partitioned table is an independent object.

In contrast, in partitioned database environments, the index is distributed across the database partitions in the same manner as the table. Because an index on a partitioned table can act independently of other indexes, some special considerations are needed with respect to which table space is used when creating an index on a partitioned table.

An index on a partitioned table is created in a single table space even if the data partitions of the table span multiple table spaces. Both DMS and SMS table spaces support the use of indexes in a different location than the table. All table spaces specified must be in the same database partition group. Each index can be placed in its own table space, including large table spaces. Each index table space must use the same storage mechanism as the data partitions, either DMS or SMS. Indexes in large table spaces can contain up to  $2^{29}$  pages.

Additional benefits of an index on a partitioned table include:

- ▶ Improved performance of drop index and online index create.
- ▶ The ability to use different values for any of the table space characteristics between each index on the table (for example, different page sizes for each index may be appropriate to ensure better space utilization).
- ▶ Reduced I/O contention providing more efficient concurrent access to the index data for the table.
- ▶ When individual indexes are dropped space will immediately become available to the system without the need for an index reorganization.
- ▶ If you choose to perform index reorganization, an individual index can be reorganized.

Figure 13-1 shows a non-partitioned index on a partitioned table residing in a single table space.



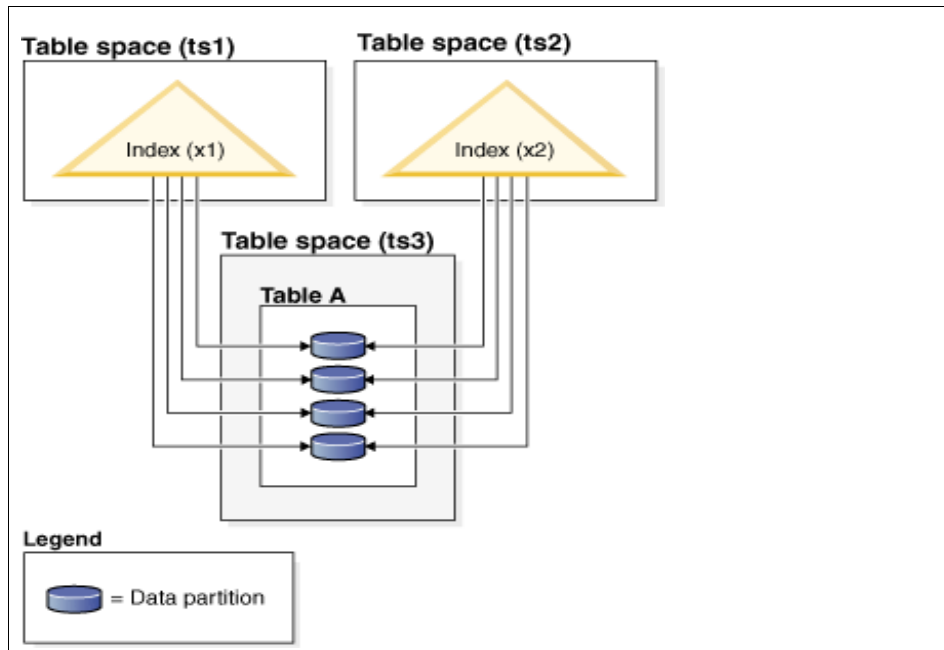


Figure 13-1 Index distribution on a partitioned table

Figure 13-2 shows index behavior on a partitioned table that is also distributed across multiple database partitions.

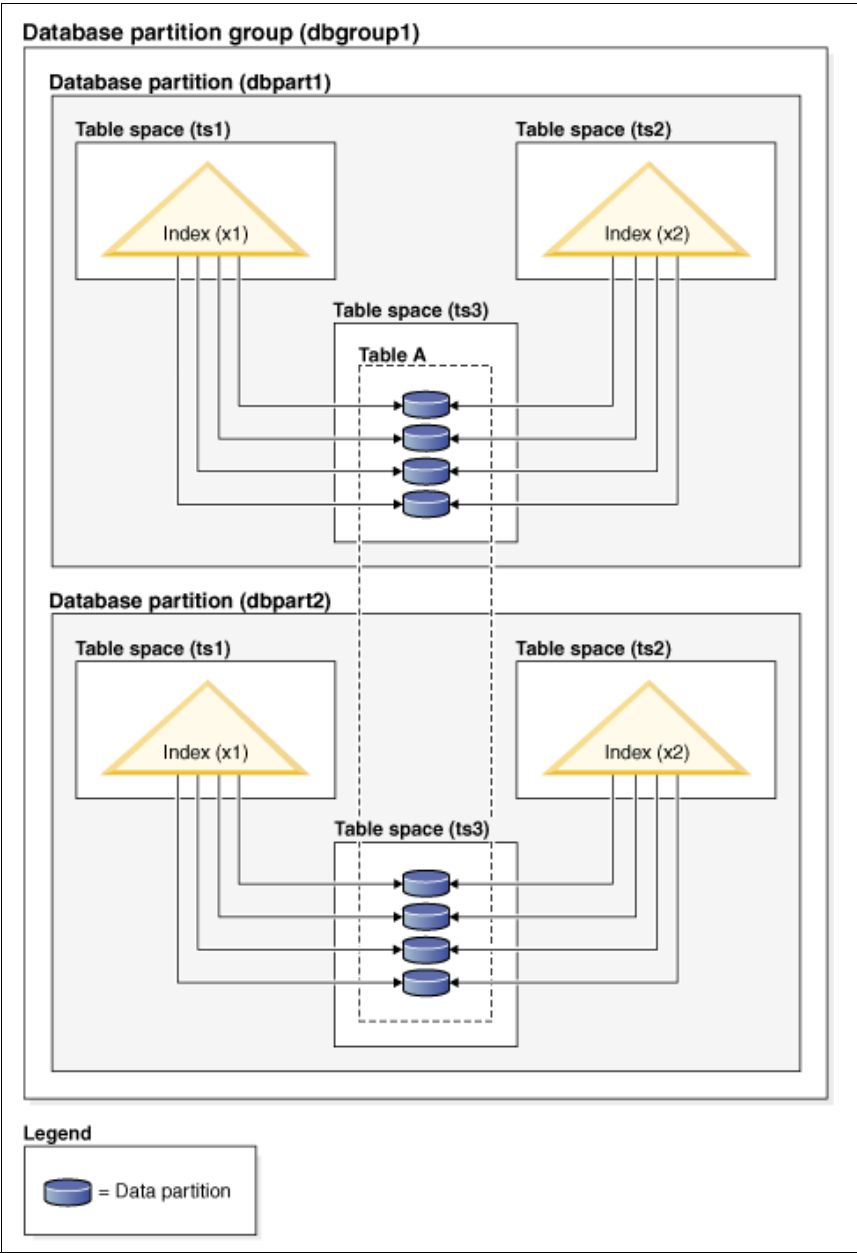


Figure 13-2 Index placement in a partitioned distributed table.

For partitioned tables only, you can override the index location by using the IN clause on the CREATE INDEX statement, which allows you to specify a table space location for the index. This approach allows you to place different indexes belonging to a partitioned table in different table spaces as required.

You can specify an index table space for a partitioned table in the CREATE INDEX ... IN <tblspace> statement, where tblspace is the name of an arbitrary table space in the database. The only restriction is that the table space must be of the same type as the data table space (that is SMS or DMS).

When you create a partitioned table without specifying where to place its non-partitioned indexes, and you create an index using the CREATE INDEX statement which does not specify a specific table space, the index is created in the table space of the first attached or visible data partition. Each of the following three possible cases is evaluated in order to determine where the index is created. The evaluation stops when there is a match to one of the cases:

- ▶ Case 1, when an index table space is specified in CREATE INDEX ... IN <tblspace1> statement, the table space specified in <tblspace1> is used for this index.
- ▶ Case 2, when an index table space is specified in the CREATE TABLE .. INDEX IN <tblspace2> statement, the table space specified in <tblspace2> is used for this index.
- ▶ Case 3, when no table space is specified, use the table space used by the first attached or visible data partition.

Where the index is created depends on what is done during the CREATE TABLE statement. For non-partitioned tables, if you do not specify any INDEX IN clause, the database fills it in for you and it is also the same as your data table space. For partitioned tables, if you leave it blank, it remains as blank, and case 3 applies.

Example 13-6 assumes the existence of a partitioned table sales (a int, b int, c int), and creates a unique index “a\_idx” in the table space “ts1”.

*Example 13-6 Creating an unique index*

---

```
CREATE UNIQUE INDEX a_idx ON sales (a) IN ts1
```

---

Example 13-7 assumes the existence of a partitioned table sales (a int, b int, c int), and creates an index “b\_idx” in the table space “ts2”.

*Example 13-7 Creating index in different table space*

---

```
CREATE INDEX b_idx ON sales (b) IN ts2
```

---

## 13.7 Approaches to defining ranges on partitioned tables

When you create a partitioned table, you can specify a range for each data partition. A partitioned table uses a data organization scheme in which table data is divided across multiple data table partitions according to the values of the table partitioning key columns of the table. Data from a given table is partitioned into multiple storage objects based on the specifications provided in the `PARTITION BY` clause of the `CREATE TABLE` statement. A range is specified by the `STARTING FROM` and `ENDING AT` values of the `PARTITION BY` clause.

To completely define the range for each data partition, you must specify sufficient boundaries. The following is a list of guidelines to consider when defining ranges on a partitioned table:

- ▶ The `STARTING` clause specifies a low boundary for the data partition range. This clause is mandatory for the lowest data partition range (although you can define the boundary as `MINVALUE`). The lowest data partition range is the data partition with the lowest specified bound.
- ▶ The `ENDING` (or `VALUES`) clause specifies a high boundary for the data partition range. This clause is mandatory for the highest data partition range (although you can define the boundary as `MAXVALUE`). The highest data partition range is the data partition with the highest specified bound.
- ▶ If you do not specify an `ENDING` clause for a data partition, then the next greater data partition must specify a `STARTING` clause. Likewise, if you do not specify a `STARTING` clause, then the previous data partition must specify an `ENDING` clause.
- ▶ `MINVALUE` specifies a value that is smaller than any possible value for the column type being used. `MINVALUE` and `INCLUSIVE` or `EXCLUSIVE` cannot be specified together.
- ▶ `MAXVALUE` specifies a value that is larger than any possible value for the column type being used. `MAXVALUE` and `INCLUSIVE` or `EXCLUSIVE` cannot be specified together.
- ▶ `INCLUSIVE` indicates that all values equal to the specified value are to be included in the data partition containing this boundary.
- ▶ `EXCLUSIVE` indicates that all values equal to the specified value are *not* to be included in the data partition containing this boundary.
- ▶ The `NULL` clause specifies whether null values are to be sorted high or low when considering data partition placement. By default, null values are sorted high. Null values in the table partitioning key columns are treated as positive infinity, and are placed in a range ending at `MAXVALUE`. If no such data partition is defined, null values are considered to be out-of-range values.

Use the NOT NULL constraint if you want to exclude null values from table partitioning key columns. LAST specifies that null values are to appear last in a sorted list of values. FIRST specifies that null values are to appear first in a sorted list of values.

- When using the long form of the syntax, each data partition must have at least one bound specified.

The ranges specified for each data partition can be generated automatically or manually.

### 13.7.1 Automatically generated ranges

Automatic generation is a simple method of creating many data partitions quickly and easily. This method is appropriate for equal sized ranges based on dates or numbers.

Example 13-8 and Example 13-9 demonstrate how to use the CREATE TABLE statement to define and generate automatically the ranges specified for each data partition.

The statement in Example 13-8 results in 12 data partitions each with 1 key value (`l_shipdate`)  $\geq$  ('1/1/1992'), (`l_shipdate`)  $<$  ('3/1/1992'), (`l_shipdate`)  $<$  ('4/1/1992'), (`l_shipdate`)  $<$  ('5/1/1992'), ..., (`l_shipdate`)  $<$  ('12/1/1992'), (`l_shipdate`)  $\leq$  ('12/31/1992').

*Example 13-8 Creating table LINEITEM*

---

```
CREATE TABLE lineitem (  
    l_orderkey    DECIMAL(10,0) NOT NULL,  
    l_quantity    DECIMAL(12,2),  
    l_shipdate    DATE,  
    l_year_month  INT GENERATED ALWAYS AS (YEAR(l_shipdate)*100 +  
    MONTH(l_shipdate)))  
    PARTITION BY RANGE(l_shipdate)  
    (STARTING ('1/1/1992') ENDING ('12/31/1992') EVERY 1 MONTH);
```

---

The starting value of the first data partition is inclusive because the overall starting bound ('1/1/1992') is inclusive (default). Similarly, the ending bound of the last data partition is inclusive because the overall ending bound ('12/31/1992') is inclusive (default). The remaining STARTING values are inclusive and the remaining ENDING values are all exclusive. Each data partition holds *n* key values where *n* is given by the EVERY clause. Use the formula (start + every) to find the end of the range for each data partition. The last data partition might have fewer key values if the EVERY value does not divide evenly into the START and END range.

In Example 13-9 there is a statement that results in 10 data partitions each with 100 key values ( $1 < b \leq 101$ ,  $101 < b \leq 201$ , ...,  $901 < b \leq 1000$ ).

*Example 13-9 Creating a table with 10 data partitions*

---

```
CREATE TABLE t(a INT, b INT)
  PARTITION BY RANGE(b) (STARTING FROM (1)
    EXCLUSIVE ENDING AT (1000) EVERY (100))
```

---

The starting value of the first data partition ( $b > 1$  and  $b \leq 101$ ) is exclusive because the overall starting bound (1) is exclusive. Similarly, the ending bound of the last data partition ( $b > 901$   $b \leq 1000$ ) is inclusive because the overall ending bound (1000) is inclusive. The remaining STARTING values are all exclusive and the remaining ENDING values are all inclusive. Each data partition holds  $n$  key values where  $n$  is given by the EVERY clause.

Finally, if both the starting and ending bound of the overall clause are exclusive, the starting value of the first data partition is exclusive because the overall starting bound (1) is exclusive. Similarly, the ending bound of the last data partition is exclusive because the overall ending bound (1000) is exclusive. The remaining STARTING values are all exclusive and the ENDING values are all inclusive. Each data partition (except the last) holds  $n$  key values where  $n$  is given by the EVERY clause.

## 13.7.2 Manually generated ranges

Manual generation creates a new data partition for each range listed in the PARTITION BY clause. This form of the syntax allows for greater flexibility when defining ranges thereby increasing your data and LOB placement options. Examples 3 and 4 demonstrate how to use the CREATE TABLE statement to define and generate manually the ranges specified for a data partition.

In Example 13-10 we see a statement that partitions on two date columns, both of which are generated. Notice the use of the automatically generated form of the CREATE TABLE syntax and that only one end of each range is specified. The other end is implied from the adjacent data partition and the use of the INCLUSIVE option.

*Example 13-10 Creating a partitioned table on two data columns*

---

```
CREATE TABLE sales(invoice_date date, inv_month int NOT NULL
GENERATED ALWAYS AS (month(invoice_date)), inv_year INT NOT
NULL GENERATED ALWAYS AS ( year(invoice_date)), item_id int NOT NULL,
cust_id int NOT NULL)
PARTITION BY RANGE (inv_year, inv_month)
(
    PART Q1_Q2 STARTING (2002,1) ENDING (2002, 3) INCLUSIVE,
    PART Q2_Q2 ENDING (2002, 6) INCLUSIVE,
    PART Q3_Q2 ENDING (2002, 9) INCLUSIVE,
    PART Q4_Q2 ENDING (2002,12) INCLUSIVE,
    PART CURRENT ENDING (MAXVALUE, MAXVALUE)
);
```

---

Gaps in the ranges are permitted. The CREATE TABLE syntax supports gaps by allowing you to specify a STARTING value for a range that does not line up against the ENDING value of the previous data partition.

Example 13-11 creates a table with a gap between values 101 and 200.

*Example 13-11 Creating a partitioned table with range gaps*

---

```
CREATE TABLE foo (a INT)
PARTITION BY RANGE(a)
(STARTING FROM (1) ENDING AT (100),
 STARTING FROM (201) ENDING AT (300))
```

---

Use of the ALTER TABLE statement, which allows data partitions to be added or removed, can also cause gaps in the ranges.

When you insert a row into a partitioned table, it is automatically placed into the proper data partition based on its key value and the range it falls within. If it falls outside of any ranges defined for the table, the insert fails and the following error is returned to the application:

```
SQL0327N The row cannot be inserted into table <tablename> because it
is outside the bounds of the defined data partition ranges.
SQLSTATE=22525
```

## 13.8 Restrictions

We can identify both table level restrictions and statement level restrictions when using data partitions.

► Table level restrictions:

Tables created using the automatically generated form of the syntax (containing the EVERY clause) are constrained to use a numeric or date time type in the table partitioning key.

► Statement level restrictions:

At the SQL statement level, we can identify the following restrictions:

- MINVALUE and MAXVALUE are not supported in the automatically generated form of the syntax.
- Ranges are ascending.
- Only one column can be specified in the automatically generated form of the syntax.
- The increment in the EVERY clause must be greater than zero.
- The ENDING value must be greater than or equal to the STARTING value.





# A

## **HACMP application server scripts**

In this appendix we provide application server scripts used in an HADR with HACMP environment. The application server start and stop scripts are used to automate the HADR takeover.

## A.1 hadr\_primary\_takeover.ksh

Example A-1 shows a sample script for HADR takeover in an HADR with HACMP environment.

*Example: A-1 hadr\_primary\_takeover.ksh*

---

```
#!/usr/bin/ksh -x

#####
# hadr_primary_takeover.ksh
# This script is called when Rotating Resource Group starts.
# Skip processes in NORMAL HACMP START UP.
# Issue hadr takeover when fail over case.
#####

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

set -x

#####
# set PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# set START_MODE
#####
# LOCAL=HOMELOCAL!=HOME
# PRE="NOMALFALLOVER(manual online operation)
# PRE!="FALLBACKFALLOVER

### /usr/es/sbin/cluster/utilities/clRGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo HACMP V5.2 must be running
### exit 1
### fi

/usr/es/sbin/cluster/utilities/clRGinfo -a
```

```

/usr/es/sbin/cluster/utilities/clRGinfo -a | grep $R_RG \
| awk -F\" '{print $2}' | awk -F\: '{print $1}' | read PRENODE

if [[ "$PRENODE" = "" ]]; then
    START_MODE="NORMAL"
else
    START_MODE="FALLOVER"
fi

#####
# MONITOR HADR ROLE & STATE
#####
#Get HADR State,HADR Role(snapshot/db cfg)
${HADR_MONITOR} ${DB2HADRINSTANCE1} ${DB2HADRINSTANCE2}
${DB2HADRDBNAME} ${VERBOSE}| read ROLE_CFG_LOCAL STATE_SNP_LOCAL
ROLE_SNP_LOCAL

#####
# MAIN
#####
case ${START_MODE} in
NORMAL)
    echo "Start trigger is Normal startup: Skip HADR operations"
;;
FALLOVER)
    echo "Start trigger is Takeover: Check HADR role & state"

    if [[ "${ROLE_SNP_LOCAL}" = "Primary" ]]; then
        # already primary
        echo "HADR role is already Primary:Skip HADR operations. When you
stop HACMP in takeover mode, stop script changes HADR role before this
script is issued."

    # elif [[ "${ROLE_SNP_LOCAL}" = "Standby" ]]; then
    #     # this instance is standby
    #     # try takeover by force
    #     echo "HADR role is Standby: Execute HADR TAKEOVER BY FORCE"
    #
    #     ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME} by force"

    elif [[ "${ROLE_SNP_LOCAL}" = "Standby" && "${STATE_SNP_LOCAL}" =
"Peer" ]]; then

```

```

        # this instance is standby peer
        # try takeover by force
        echo "HADR role is Standby and HADR status is Peer: Execute HADR
TAKEOVER BY FORCE:"
        ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME} by force"

        elif [[ "${ROLE_SNP_LOCAL}" = "Standby" && "${STATE_SNP_LOCAL}" !=
"Peer" ]]; then
            # this instance is not standby peer
            echo "HADR role is Standby but HADR status is Not Peer: You might
lose data by takeover for inconsistency of databases, Skip HADR
operations."

        else
            echo "HADR role is neithe Primary or Standby: Skip HADR
operations "

        fi
    ;;

esac

date

exit 0

```

---

## A.2 hadr\_primary\_stop.ksh

Example A-2 shows a sample stop script for HADR takeover in an HADR with HACMP environment.

*Example: A-2 hadr\_primary\_stop.ksh*

---

```

# Normal stop operation: No operation is done.
# Stop in takeover mode: If HADR Primary & Peer, execute hadr takeover
on Standby by remote command
#####

exec >> /tmp/`basename $0`.log
exec 2>&1

echo "#####"
date
echo "#####"

```

```

set -x

#####
# set PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# set STOP_MODE
#####
# LOCAL=HOMELOCAL!=HOME
# PRE=""NOMALFALLOVER(manual online operation)
# PRE!=""FALLBACKFALLOVER

### /usr/es/sbin/cluster/utilities/clRGinfo -a >/dev/null 2>&1
### if [[ $? != 0 ]]; then
### echo HACMP V5.2 must be running
### exit 1
### fi

/usr/es/sbin/cluster/utilities/clRGinfo -a

/usr/es/sbin/cluster/utilities/clRGinfo -a | grep $R_RG \
| awk -F\" '{print $2}' | awk -F\: '{print $2}' | read POSTNODE

if [[ "$POSTNODE" = "" ]]; then
    STOP_MODE="NORMAL"
else
    STOP_MODE="FALLOVER"
fi

#####
# MONITOR HADR ROLE & STATE
#####
#Get HADR State, HADR Role (snapshot, db cfg)
${HADR_MONITOR} ${DB2HADRINSTANCE1} ${DB2HADRINSTANCE2}
${DB2HADRDBNAME} ${VERBOSE} | read ROLE_CFG_LOCAL STATE_SNP_LOCAL
ROLE_SNP_LOCAL

#####
# MAIN

```

```
#####
case ${STOP_MODE} in
NORMAL)
    echo "Stop HACMP in graceful mode"
;;
FALLOVER)
    echo "Stop HACMP in takeover mode "

    if [[ "${ROLE_SNP_LOCAL}" = "Primary" && "${STATE_SNP_LOCAL}" =
"Peer" ]]; then
        echo "HADR takeoveris issued on remote node"
        /usr/es/sbin/cluster/sbin/cl_nodecmd -cspoc "-n ${REMOTENODE}"
        ${SU_CMD} "db2 takeover hadr on db ${DB2HADRDBNAME}"
    fi
;;

esac

date

exit 0
```

env.sh

```
DB2HADRINSTANCE1=hadrinst      # replace db2instp with P's instance
name
DB2HADRINSTANCE2=hadrinst      # replace db2insts with S's instance
name
DB2HADRDBNAME=sample           # replace hadrdb with HADR db name
VERBOSE=verbose                # change to verbose to get more
diagnostics logged
R_RG=hadr_rg                   rotating resource group name
C_RG=db2_rg                    concurrent resource group name
```

```
SU_CMD="su - ${DB2HADRINSTANCE1} -c"          #su command
LOCALNODE=~ /usr/es/sbin/cluster/utilities/get_local_nodename`
#LOCAL NODE
#REMOTENODE=~ /usr/es/sbin/cluster/utilities/clrsctinfo -cp cllsif |
grep -v ${LOCALNODE} | cut -f 6 -d ":" | uniq`    #REMOTE NODE
```

LANG=C

```

/usr/bin/env LANG=C /usr/es/sbin/cluster/utilities/clshowres -g
"${R_RG}" | grep "Participating Node Name(s)" | awk '{print $4 " " $5}'
| read n1 n2

if [[ "${n1}" = "${LOCALNODE}" ]];then
    REMOTENODE=${n2}
else
    REMOTENODE=${n1}
fi

HADR_MONITOR="/home/hadrinst/scripts/hadr_monitor.ksh"      # Get HADR
State HADR Role (snapshot, db cfg)
#!/usr/bin/ksh

```

---

## A.3 hadr\_monitor.ksh

Example A-3 shows a sample script for monitoring the HADR state.

*Example: A-3 hadr\_monitor.ksh*

---

```

# hadr_monitor.ksh
#####
# hadr_monitor.ksh
# This script is called by Concurrent Resource Group start script,
# Rotate Resource Group starts/stops script,
# and application monitor.
#
# 1.Get HADR role and state by database snapshot
# 2.Get HADR role by database configurations
#####
set -x

#####
# SET PARAMETER
#####
. /home/hadrinst/scripts/env

#####
# GET HADR ROLE & STATE
#####

```

```

HADR_PROBE=~${SU_CMD} "db2 get snapshot for database on
${DB2HADRDBNAME}" | awk '($1 == "State" && $2 == "=") || ($1 == "Role"
&& $2 == "=")' | sort | awk '{print $3}'`
echo $HADR_PROBE | read hadr_role hadr_state other

if [[ "${hadr_state}" = "" || "${other}" != "" ]];then
    hadr_role=$( ${SU_CMD} "db2 get snapshot for database on
${DB2HADRDBNAME}" | awk ' $1 == "Role" && $2 == "=" {print $3}' )
    hadr_state=$( ${SU_CMD} "db2 get snapshot for database on
${DB2HADRDBNAME}" | awk ' $1 == "State" && $2 == "=" {print $3}' )
fi

hadr_role_cfg=$( ${SU_CMD} "db2 get db cfg for ${DB2HADRDBNAME} | grep
'HADR database role' | awk '{print \$5}')"

echo $hadr_role_cfg $hadr_state $hadr_role

exit 0

```

---



# Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this book.

## IBM Redbooks

For information about ordering these publications, see “How to get IBM Redbooks” on page 414. Note that some of the documents referenced here may be available in softcopy only.

- ▶ *WebSphere Information Integrator Q Replication: Fast Track Implementation Scenarios*, SG24-6487
- ▶ *DB2 Integrated Cluster Environment Deployment Guide*, SG24-6376

## Other publications

These publications are also relevant as further information sources:

### IBM - DB2 9

- ▶ *What's New*, SC10-4253
- ▶ *Administration Guide: Implementation*, SC10-4221
- ▶ *Administration Guide: Planning*, SC10-4223
- ▶ *Administrative API Reference*, SC10-4231
- ▶ *Administrative SQL Routines and Views*, SC10-4293
- ▶ *Call Level Interface Guide and Reference, Volume 1*, SC10-4224
- ▶ *Call Level Interface Guide and Reference, Volume 2*, SC10-4225
- ▶ *Command Reference*, SC10-4226
- ▶ *Data Movement Utilities Guide and Reference*, SC10-4227
- ▶ *Data Recovery and High Availability Guide and Reference*, SC10-4228
- ▶ *Developing ADO.NET and OLE DB Applications*, SC10-4230
- ▶ *Developing Embedded SQL Applications*, SC10-4232
- ▶ *Developing Java Applications*, SC10-4233

- ▶ *Developing Perl and PHP Applications*, SC10-4234
- ▶ *Getting Started with Database Application Development*, C10-4252
- ▶ *Getting started with DB2 installation and administration on Linux and Windows*, GC10-4247
- ▶ *Message Reference Volume 1*, SC10-4238
- ▶ *Message Reference Volume 2*, SC10-4239
- ▶ *Migration Guide*, GC10-4237
- ▶ *Performance Guide*, SC10-4222
- ▶ *Query Patroller Administration and User's Guide*, GC10-4241
- ▶ *Quick Beginnings for DB2 Clients*, GC10-4242
- ▶ *Quick Beginnings for DB2 Servers*, GC10-4246
- ▶ *Spatial Extender and Geodetic Data Management Feature User's Guide and Reference*, SC18-9749
- ▶ *SQL Guide*, SC10-4248
- ▶ *SQL Reference, Volume 1*, SC10-4249
- ▶ *SQL Reference, Volume 2*, SC10-4250
- ▶ *System Monitor Guide and Reference*, SC10-4251
- ▶ *Troubleshooting Guide*, GC10-4240
- ▶ *Visual Explain Tutorial*, SC10-4319
- ▶ *XML Extender Administration and Programming*, SC18-9750
- ▶ *XML Guide*, SC10-4254
- ▶ *XQuery Reference*, SC18-9796
- ▶ *DB2 Connect User's Guide*, SC10-4229
- ▶ *Quick Beginnings for DB2 Connect Personal Edition*, GC10-4244
- ▶ *Quick Beginnings for DB2 Connect Servers*, GC10-4243

## **IBM - DB2 8.2**

- ▶ *What's New V8*, SC09-4848-01
- ▶ *Administration Guide: Implementation V8*, SC09-4820-01
- ▶ *Administration Guide: Performance V8*, SC09-4821-01
- ▶ *Administration Guide: Planning V8*, SC09-4822-01
- ▶ *Application Development Guide: Building and Running Applications V8*, SC09-4825-01

- ▶ *Application Development Guide: Programming Client Applications V8*, SC09-4826-01
- ▶ *Application Development Guide: Programming Server Applications V8*, SC09-4827-01
- ▶ *Call Level Interface Guide and Reference, Volume 1, V8*, SC09-4849-01
- ▶ *Call Level Interface Guide and Reference, Volume 2, V8*, SC09-4850-01
- ▶ *Command Reference V8*, SC09-4828-01
- ▶ *Data Movement Utilities Guide and Reference V8*, SC09-4830-01
- ▶ *Data Recovery and High Availability Guide and Reference V8*, SC09-4831-01
- ▶ *Guide to GUI Tools for Administration and Development*, SC09-4851-01
- ▶ *Installation and Configuration Supplement V8*, GC09-4837-01
- ▶ *Quick Beginnings for DB2 Clients V8*, GC09-4832-01
- ▶ *Quick Beginnings for DB2 Servers V8*, GC09-4836-01
- ▶ *Replication and Event Publishing Guide and Reference*, SC18-7568
- ▶ *SQL Reference, Volume 1, V8*, SC09-4844-01
- ▶ *SQL Reference, Volume 2, V8*, SC09-4845-01
- ▶ *System Monitor Guide and Reference V8*, SC09-4847-01
- ▶ *Data Warehouse Center Application Integration Guide Version 8 Release 1*, SC27-1124-01
- ▶ *DB2 XML Extender Administration and Programming Guide Version 8 Release 1*, SC27-1234-01
- ▶ *Federated Systems PIC Guide Version 8 Release 1*, GC27-1224-01

## **HACMP**

- ▶ *HACMP v5.3 Administration Guide* SC23-4862-06
- ▶ *HACMP v5.3 Concepts and Facilities Guide* SC23-4864-06
- ▶ *HACMP v5.3 Planning and Installation Guide* SC23-4861-06
- ▶ *HACMP v5.4 Master Glossary*, SC23-4867-08

## **Online resources**

These Web sites are also relevant as further information sources:

- ▶ DB2 Information Center  
<http://publib.boulder.ibm.com/infocenter/db2luw/v9/index.jsp>

## How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

[ibm.com/redbooks](http://ibm.com/redbooks)

## Help from IBM

IBM Support and downloads

[ibm.com/support](http://ibm.com/support)

IBM Global Services

[ibm.com/services](http://ibm.com/services)

# Index

## Symbols

.rhosts 371

## A

access permission 161  
ACR xi, 115  
activate database 31, 283  
active log 211  
active primary node 19  
active/active 188  
active/standby 188  
administrative view 103  
algorithm 217  
Algorithm for Recovery and Isolation Exploiting  
Semantics (ARIES) 28  
alternate server 66, 116  
application server 185  
archive log 211  
array 364  
ASN 328  
ASYNC 44  
asynchronous 9  
asynchronous mode 35, 40  
asynchronous page cleaner 213  
automatic client reroute xi, 115  
automatic client reroute (ACR) 26  
automatic client reroute (acr) 47  
AUTORESTART 85  
availability 10

## B

balanced configuration unit 373  
basic operation 68  
batch job 213  
BCU 373  
bidirectional 16  
BLOBs 95  
bottleneck 366  
buffer pool 28, 30, 42, 94  
buffer size 143  
BY FORCE option 77

## C

capture program 14  
catalog partition 363  
catalog table space 210  
catch-up phase 36  
catchup phase 32  
circular logging 43  
client reroute 119  
CLOBs 95  
cluster 11, 232, 367  
cluster server concept 216  
clustering 10  
clustering solution 10  
command line xi, 41, 65  
Command Line Processor (CLP) 44, 166  
command line steps 88  
commit statement 145  
compression 147  
configuration 43, 48  
    target to source server connectivity 349  
    using Setup HADR wizard 46  
Configuration Assistant 54  
congestion 143  
connection interval 117  
connection status 100  
context object 132  
Control Center 69  
coordinator node 361  
creating a cluster server 220  
creating a DB2 group 240  
creating a DB2 instance 235  
cross-partition join 364  
cursors 123

## D

daemon 367  
dark fiber 9  
data definition language (DDL) 94  
data links 95  
data loss 78  
data manipulation language (DML) 94  
data partition 384  
data source movement 25

- database
  - backup 97
  - restore 97
- database alias 372
- database configuration 95
- database configuration parameter
  - AUTORESTART 138
  - CHNGPGS\_THRESH 213
  - HADR\_DB\_ROLE 139
  - HADR\_LOCAL\_HOST 139
  - HADR\_LOCAL\_SVC 140
  - HADR\_REMOTE\_HOST 140
  - HADR\_REMOTE\_INST 140
  - HADR\_REMOTE\_SVC 140
  - HADR\_SYNCMODE 140
  - HADR\_TIMEOUT 141
  - INDEXREC 141
  - LOGARCHMETH1 62
  - LOGARCHMETHn 139
  - LOGBUFSZ 32
  - LOGFILSIZ 142
  - LOGINDEXBUILD 139
  - LOGRETAIN 66, 139
  - NUM\_IOCLEANERS 212
  - SELF\_TUNING\_MEM 142
  - SOFTMAX 213
  - SYSADM\_GROUP 81
- database managed space 209
- database manager configuration parameter 372
- database partition 359–360, 363
- datasource 121, 131
- datasource object 132
- DB parameter 43
- DB2
  - configuration parameters 181
  - FixPak rolling upgrades 166
  - system upgrades 165
  - version upgrade 173
- DB2 Admin Server (DAS) 47
- DB2 engine 29
- DB2 FixPak
  - using command line 172
  - Windows 166
- DB2 HADR 24
- DB2 logical log architecture 28
- DB2 registry variable
  - DB2\_CONNRETRIES\_INTERVAL 120
  - DB2\_HADR\_BUF\_SIZE 143
  - DB2\_MAX\_CLIENT\_CONNRETRIES 120
  - DB2LOADREC 144
  - DB2TCP\_CLIENT\_CONTIMEOUT 120
  - LOAD\_COPY\_NO\_OVERRIDE 144
- DB2 with Microsoft Windows Cluster Server 215
- DB2\_GRP\_LOOKUP 81
- db2agent 29
- db2clini.ini 121
- db2diag.log filtering output 112
- db2gcf 205
- db2hadrp 31–32
- db2hadrs 31–32
- db2icrt 236
- db2ilist 236
- db2lfr 32
- db2loggr 30
- db2loggw 30, 32
- db2nodes.cfg 203, 369
- db2pd 83
- db2start command 68
- db2support command 177
- DB2TCP\_CLIENT\_RCVTIMEOUT 120
- DBM CFG parameter 43
- deactivate database 285
- deactivate database command 31, 71, 174
- dirty page 212
- disaster recovery 20
- disk array 9
- disk controller 5, 373
- disk mirroring 5
- disk shared cluster 18
- disk striping 5
- distribution key 360–361, 364
- distribution map 364
- dll 218
- DNS 83
- domain 220, 223
- Dynamic Link Library (dll) 218

**E**

- Engine Dispatch Unit (EDU) 30
- enhanced concurrent volume group 211
- Enterprise Server Edition (ESE) 42
- ethernet 208

**F**

- failback 18, 37
- failover 24, 36–37, 373
- failover script 380

- failure 37
- failure detection 25
- Fast Communications Manager 370
- FCM 370
- federated processing 123
- file system 367
- firewall 34
- FlashCopy 159
- flush 148
- FORCE 34
- fsck 211
- FTP 55

## G

- get db cfg 83
- GET SNAPSHOT command 167
- GET SNAPSHOT FOR DATABASE command 98
- get snapshot for db 83
- global temporary table 123
- Graphical User Interface (GUI) xi, 41
- groups 218
- GUI HADR Setup wizard 44

## H

- HA clustering software (HACMP) 24
- HADR SYNCMODE 44
- HADR takeover command 34
- HADR\_TIMEOUT 39
- hash value 364
- hashing algorithm 364
- heart beat 27, 145
- heartbeat 31, 100
- high availability 4
- high availability (HA)
  - continuous availability 25
  - failover availability 25
  - overview 24
  - types 25
- High Availability Cluster Multi-Processing 183
- High Availability Disaster Recovery (HADR) 23–24
  - administrating systems 94
  - administration and monitoring xi, 93
  - architecture 30
  - benefits 27
  - communication 27
  - configuration 43
  - considerations 85
  - handling failure 34

- monitoring
  - administrative view and table function 103
  - db2diag 107
  - db2pd 101
  - overview 25
  - primary database 32
  - re-establishing after failure 86
  - replication 34
  - setup xi, 41
  - shutdown 68
  - standby database 33
  - startup 68
  - synchronization modes 35–36
  - topology 34
- high-speed switch 371
- home directory 199
- home file system 367
- hosts.equiv 205
- Hot Standby 379
- hybrid 18

## I

- I/O backplane 373
- I/O path 373
- identity 123
- Idle Standby 379
- indoubt 138
- inflight 138
- in-flight transaction 145
- installing DB2 ESE 231
- instance 203, 246
- instance node name 51
- integrity check 209
- inter-partition 373
- IP address 242
- IP address redirection 26

## J

- JCC 118
- JNDI 132

## K

- keepalive 186
- kernel 148

## L

- LEAD 46

- license key 198
- linear/archival logging 62
- LOAD COPY NO 97
- LOAD COPY YES 96
- LOAD utility 160
- local area network (LAN) 44
- local standby node 20
- log archive 162
- log archive device 162
- log file 14, 147
- Log File Reader (LFR) 30
- log files 98
- log gap running average 100
- log position 100
- log reader 30
- log record 30
- log sequence number (LSN) 97
- log shipping mode 145
- logical data group 103
- logical log 32
- logical log buffer (llb) 28
- logical log record (LLR) 28
- logical name 132
- logical unit of work 28
- logical volume 197
- LOGINDEXBUILD 62, 66

## M

- main script 185
- majority node set (MNS) 217
- manual intervention 116
- manually configuring DB2 instance in Windows Cluster 237
- maxRetriesForClientReroute 121
- memory 367
- message broker 14
- message queue 13
- metadata 363
- Microsoft Cluster Server definitions 218
- Microsoft Cluster Servers (MSCS) 216
- Microsoft Management Console Services Snap-in 168
- migrate database command 177
- migrating the DB2 instance to the cluster environment 244
- MINCOMMIT 30
- minimum configuration and sample setup 219
- mirroring 9

- monitor element 103
- monitor element output 101
- monitoring
  - administrative view and table function 103
  - db2diag 107
  - db2pd 101
  - summary 113
- MQSC command line 320
- multi-partitioned database 360
- multiple processor 360
- Mutual Takeover 380

## N

- native client library 130
- NEAR SYNCHRONOUS mode 35
- NEARSYNC 44
- near-sync mode 39
- NetBIOS 53
- netname 371
- network card 139
- network file system 161
- Network Load Balance 216
- network parameter 119
- NFS 161, 367
- NFSDs 367
- NLB 216
- node 10, 359, 364
- node configuration file 369
- non logged large object 95
- nonrecoverable load 97
- non-replicated operation 95

## O

- offline backup 177
- offline REORG 94
- OLAP 358
- OLTP 7, 358
- online analytical processing 358
- online REORG 94
- online transaction processing 358
- operating system (OS) 181
- out of band 38
- outage 177
- outage period 38

## P

- packet 149



- page cleaner 212
- paging space 367
- parallel system 357
- parameter 367
- partition group 363, 373
- partitioning key column 385
- passive node 20
- Peer state 32, 38, 78, 99
- peer to peer 16
- performance 10, 373
- physical node 192
- planned takeover 73
- POLONIUM 46
- port number 140
- primary 37
- primary and standby systems 25
- primary database 17, 38
  - reintegration 79
- primary reintegration 38, 40
- property 121
- protocol 123

## Q

- Q Apply Control table
  - creation 327
- Q Capture Control table 316
  - creation 322
- Q Capture program 316
- Q replication xii, 13, 315
  - bidirectional 316
  - introduction 316
  - peer-to-peer 316
  - types 316
  - unidirectional 316
- Q subscription 317
  - creation 331
- quorum 217
- Quorum cluster server 216
- quorum cluster server 217

## R

- RAID 9
- RAID 0 5
- RAID controller 5
- range 384
- raw device 184
- receiving socket 148
- recovery history file 95

- Redbooks Web site 414
  - Contact us xv
- redundancy 22
- redundant 116
- redundant array of independent disks 4–5
- registry variable 119, 143
- remote catchup 99
- remote catchup pending 99
- remote command 371
- remote shell 205
- replicating LOAD operations 96
- replication architecture 14
- replication center 351, 353
- replication solution 13
- repository 134
- resource 218
- resource group 20, 184, 193
- resource type 218, 237
- response file 369
- retryIntervalForClientReroute 121
- role 99
- rolling upgrade 18
- rsh 205

## S

- scalability 366
- SCSI 223
- secure shell 205
- security risk 371
- send queue 316
- sequences 123
- server option 123
- service address 184
- service name 140
- service node 194
- session resource 123
- setup xi
  - initial database source and target 318
  - preparing the environment 43
  - recommendations 42
  - requirements 42
  - WebSphere MQ objects on source server 318
  - WebSphere MQ objects on target server 319
- shared disk 185, 218
- shutdown 68, 71
  - from the Control Center 72
  - using command 71
- single partition 203

- single quorum servers 217
- single-partition database 360
- sleep time 120
- SMP server 357
- snapshot 98, 159
- soft checkpoint 213
- software-based replication 13
- source system 14
- spare disk 5
- split brain 34, 39, 85
- SQL queries 104
- SQL Replication 24
- SQL replication 13
- SQL1018N error 52
- SQL30081N Communication error 83
- ssh 205
- staging environment 22
- standard (database) 39
- standard mode 34
- standby database 17, 31, 38
- standby node 197, 203, 209
- start hadr 283
- START HADR AS PRIMARY 97
- START HADR command 31, 71
- starting listener and channel on source and target 320
- starting Q Apply 352–353
- starting Q Capture 350
- startup 68
- state 99
- STOP HADR 71
- Storage Area Network (SAN) 217
- Storage Copy 159
- Storage Enclosure 9
- storage object 384
- stored procedures and UDFs 95
- SUSE Linux 10 43
- switch role 90
- sync mode 39
- synchronization mode 147
- synchronous 9
- synchronous mode 35

## T

- table collocation 361
- table function 103
- table space 94, 363
- tablespace container 28

- takeover 34, 40
- takeover by force 77
  - GUI 79
- takeover command 73
- takeover hadr command 73
- target system 13
- TCP/IP 367
- tcp\_keepinit 119
- tcp\_recvspace 148
- temporary table spaces 210
- terminology 36
- testing the queues 321
- thread 29
- Tivoli System Automation (TSA) 24
- tmscsi 208
- token 208
- topology 34, 184
- transaction 28
- transactional integrity 28
- tree structure 54
- troubleshooting 80
  - after HADR disconnect or server failure 84
  - after setup or during normal execution 82
  - during setup 81
- Type 4 connectivity 121
- types of clusters 216

## U

- unidirection Q replication
  - setup 318
- unidirectional 16
- unidirectional setup 317
- units of work 28
- user table spaces 210
- using GUI 74

## V

- varyon 159
- vector 364
- version upgrade 173, 179
  - Windows 174
- virtual machine 10
- volume group 159

## W

- WebSphere Information Integrator Q Replication 24

WebSphere MQ messages queues 316  
Windows version upgrade 174  
wireless area network (WAN) 44  
workload 10, 361  
write ahead logging (WAL) protocol 28

## **X**

X11 emulator 44



# Abbreviations and acronyms

<b>ACR</b>	Automatic client reroute	<b>LSN</b>	Log sequence number
<b>BCU</b>	Balanced configuration unit	<b>LSN</b>	log sequence number
<b>BP</b>	Buffer pool	<b>MNS</b>	Majority node set
<b>BPU</b>	Balanced partition unit	<b>MSCS</b>	Microsoft Cluster Server
<b>CLP</b>	Command Line Processor	<b>NLB</b>	Network Load Balance
<b>CS</b>	Cursor stability	<b>OLAP</b>	Online analytical processing
<b>DAS</b>	DB2 Administration Server	<b>OLTP</b>	Online Transaction Processing
<b>DDL</b>	Data Definition Language	<b>OS</b>	Operating system
<b>dll</b>	Dynamic Link Library	<b>PPRC</b>	Peer to Peer Remote Copy
<b>DML</b>	Data Manipulation Language	<b>RAID</b>	Redundant array of independent disks
<b>DMS</b>	DB2 database managed space	<b>rsh</b>	remote shell
<b>DNS</b>	Domain Name Server	<b>SAN</b>	Storage Area Network
<b>DPF</b>	Database Partitioning Feature	<b>scp</b>	Secure Shell Copy
<b>ECM</b>	Enhanced Concurrent Volume Groups	<b>ssh</b>	Secure shell
<b>EDU</b>	Engine Dispatch Unit	<b>STMM</b>	Self Tuning Memory Manager
<b>ESE</b>	Enterprise Server Edition	<b>TSA</b>	Tivoli System Automation
<b>FCM</b>	Fast Communication Manager	<b>VG</b>	Volume group
<b>GUI</b>	Graphical User Interface	<b>WAL</b>	Write ahead logging
<b>HA</b>	High availability	<b>WAS</b>	WebSphere Administration Server
<b>HACMP</b>	High Availability Cluster Multi-Processing		
<b>HADR</b>	High Availability and Disaster Recovery		
<b>HSM</b>	Hierarchical Storage Management		
<b>JNDI</b>	Java Naming and Directory Interface		
<b>LFR</b>	Log File Reader		
<b>LLB</b>	Logical Log Buffer		
<b>LLR</b>	Logical Log Record		
<b>LOBs</b>	Logged Large Objects		





## High Availability and Scalability Guide for DB2 on Linux, UNIX, and Windows

(0.5" spine)  
0.475" <-> 0.875"  
250 <-> 459 pages









# High Availability and Scalability Guide for DB2 on Linux, UNIX, and Windows



**Learn HADR setup, administration, monitoring, and best practices**

**Use HACMP, TSA, and MSCS with DB2 and DB2 HADR**

**Quick start with DB2 DPF**

As organizations strive to do more with less, IBM DB2 for Linux, UNIX, and Windows provides various built-in high availability features. DB2 further provides high availability solutions by leveraging enterprise system resources with broad support for clustering software such as HACMP, TSA, and Microsoft Windows Cluster Server.

This IBM Redbooks publication describes DB2's high availability functions and features, focusing on High Availability Disaster Recovery (HADR) in the OLTP environment. The book provides a detailed discussion of HADR, including setup, configuration, administration, monitoring, and best practices.

We explain how to configure cluster software HACMP, TSA, and MSCS with DB2 and show how to use these products to automate HADR takeover.

DB2 also provides unprecedented enterprise-class scalability to 32-bit and 64-bit Linux, UNIX, and Windows platforms. We introduce the DB2 Database Partitioning Feature, covering the installation, configuration, and scaling of a database partition.

## INTERNATIONAL TECHNICAL SUPPORT ORGANIZATION

### BUILDING TECHNICAL INFORMATION BASED ON PRACTICAL EXPERIENCE

IBM Redbooks are developed by the IBM International Technical Support Organization. Experts from IBM, Customers and Partners from around the world create timely technical information based on realistic scenarios. Specific recommendations are provided to help you implement IT solutions more effectively in your environment.

**For more information:**  
[ibm.com/redbooks](http://ibm.com/redbooks)